Vehicloak: A Blockchain-Enabled Privacy-Preserving Payment Scheme for Location-based Vehicular Services

Yihao Guo, Zhiguo Wan, Hui Cui, Xiuzhen Cheng, Falko Dressler

Abstract—The Internet of Vehicles (IoV) technology enables vehicles to communicate with each other, with pedestrians and with roadside infrastructures, to realize more efficient, safer and more environmentally friendly transportation. IoV also promises rich location-based services for vehicles, such as parking and toll highway. However, preserving privacy for location-based service payments emerges as a critical and challenging problem in IoV. Existing schemes rely on centralized banks for payment processing, resulting in location privacy leakage to centralized entities.

In this paper, we utilize blockchain as the payment method, and propose a decentralized privacy-preserving payment scheme named Vehicloak for IoV. The biggest challenge is to provide location privacy for vehicles while guaranteeing correct service payments using the transparent blockchain. To tackle this challenge, we introduce a new cryptographic technique called zk-GSigproof that integrates zero-knowledge proof with group signature. Vehicloak deploys this technique in a smart contract to process payment, which verifies zero-knowledge proof and group signature without leaking location information. It is not limited to IoV and can be applied in many payment scenarios. To evaluate the performance of our scheme, we implement Vehicloak on a private blockchain of 100 nodes on Aliyun, and conduct a test with up to 4,000 transactions. The experimental results prove the feasibility of Vehicloak.

Index Terms—Blockchain, zk-SNARK, Group signature, Location privacy, Smart contract.

1 Introduction

The Internet of Vehicles (IoV) technology employs big data, cloud computing, artificial intelligence and other techniques to realize pervasive network connections and data exchanges among vehicles, roads and pedestrians, which gives rise to an integrated system that is capable of dynamic information services, intelligent vehicle control and traffic management. In addition, IoV has the potential to bring great benefits in improving traffic jams and reducing harmful exhaust gas and traffic accidents [1].

Unfortunately, IoV involves severe security threats for vehicles in addition to the aforementioned great benefits. It has caused countless economic losses due to security and privacy attacks. The vehicle itself can be the target of adversaries, while centralized databases of vehicle information can also be attacked due to the high value of such information. Even worse, most vehicle service providers collect a large amount of personal information, and employ advanced information technologies such as big data analysis and artificial intelligence to gain valuable knowledge about their clients. This would lead to potential centralization, monopoly, and severe privacy leak [2], [3].

The protection of vehicle location-based privacy emerges as a critical problem in recent years. To realize intelligent traffic management, vehicles have to periodically report their identities as well as location and driving status information to a centralized server. Vehicular services like parking toll and electronic toll collection (ETC) require location information of vehicles to determine the correct charges [2]. Without appropriate protection on vehicle location privacy, it is easy for an adversary to track vehicles through data analytics. This problem becomes even more challenging when location information is required for payment processing. For example, an ETC system calculates the toll cost based on the entry and exit points of a given vehicle, and the payment process would expose the relationship between the vehicle driver's identity and the entry/exit points.

Recently, the blockchain technology has attracted tremendous interests from government and academia to industry for its decentralization, transparency and immutability [4]. It can solve the traditional single point of failure problem, achieve effective access control, and even enable mutually distrustful parties to establish trust relationship in wireless networks [5]. Hence, blockchain is quickly introduced to many areas including finance [6], vaccine [7], smart grid [2], UAV swarms [8], [9] and cloud services [10]. However, as a decentralized and transparent system, a delicate mechanism should be designed to preserve privacy during the payment process [11].

In this paper, we attempt to address this problem and design a blockchain-based privacy-preserving payment scheme for location-based services. For the sake of conve-

Corresponding author: Zhiguo Wan.
Y. Guo and X. Cheng are with the School of Computer Science and Technology, Shandong University, Qingdao, Shandong, P.R. China (e-mail:{ yhguo@mail., xzcheng@}sdu.edu.cn).
Z. Wan is with Zhejiang Lab, Hangzhou, Zhejiang, China (email:wanzhiguo@zhejianglab.com).
H. Cui is with the Discipline of Information Technology, Mathematics, and Statistics, Murdoch University, Perth, WA 6150, Australia, and also with the Data61, CSIRO, Melbourne, VIC 3008, Australia (e-mail: hui.cui@murdoch.edu.au).
F. Dressler is with the School of Electrical Engineering and Computer Science, TU Berlin, Germany (e-mail: dressler@ccs-labs.org).

nience, we highlight our contributions as follows:

1) We design Vehicloak, a blockchain-based privacy-preserving payment scheme for vehicular location-based services in IoV. To the best of our knowledge, Vehicloak is the first blockchain-based scheme that preserves location privacy for vehicle payment. A key component within Vehicloak is a new cryptographic technique zk-GSigproof, which preserves privacy while guaranteeing correct payment amount. Besides that, Vehicloak has good universality and can be applied to multiple application scenarios such as electricity billing and parking toll.
2) We formulate a rigorous security definition for zk-GSigproof, and formally demonstrate its security. In addition, we provide an in-depth analysis on security and privacy of Vehicloak, and give a comprehensive discussion on a few practical considerations. According to our analysis, one can see that Vehicloak not only can thwart existing attacks targeting vehicle location privacy, but also demonstrates high feasibility.
3) We fully implement Vehicloak on a customized private Ethereum blockchain (BlockMaze [4])—including zk-GSigproof and the smart contract Contract-Vehicloak. For BlockMaze, we adopt the one-time pseudonym technique to achieve privacy-preserving interactions between vehicles and smart contracts. Finally, we conduct comprehensive experiments to evaluate the performance of Vehicloak, and the results show that Vehicloak is highly efficient in processing payments on the blockchain.

The remainder of the paper is structured as follows: We first review the most related works in Section 2; then describe the system and threat models, and briefly introduce the necessary preliminary knowledge in Section 3. After that, we propose a novel technique zk-GSigproof and provide a security analysis. In Section 5, we detail our scheme Vehicloak. Section 6 describes the simulation experiments on Aliyun to evaluate the performance of Vehicloak. Finally, we conclude this paper with a future research discussion.

2 Related Works

Research works related to Vehicloak include the payment scenarios in IoV and vehicular location-based privacy-preserving services. According to their design ideas, we categorize them into traditional centralized schemes and blockchain-based distributed ones.

2.1 Traditional Centralized Schemes

VPriv [12] realizes location privacy protection based on an out-of-band enforcement mechanism, which combines techniques such as homomorphic encryption and random spot checks. PrETP [13] makes use of a new cryptographic protocol named Optimistic Payment, which only discloses a small amount of data to prove the correctness of the payable amount. Based on VPriv and PrETP, Meiklejohn et al. [14] proposed Milo. Its key techniques include zero-knowledge proofs and blind identity-based encryption. All the above solutions are based on Global Navigation Satellite Systems (GNSS), in which on-board units play the role of centralized data collection and processing. Unlike the schemes [12]–[14], P4TC [15] employs Dedicated Short-Range Communication (DSRC) to realize privacy-preserving toll collection. Hu et al. [3] developed PPDIR, which achieves privacy preservation and billing via delayed information release.

The schemes mentioned above aim to achieve privacy protection for vehicle information, especially location information. However, these schemes assume that the third party is trusted in processing data and payment. In practice, the third party is not always trustworthy—it may leak private information and may be vulnerable to single-point-of-failures. In addition, realizing privacy-preserving payments in IoV is challenging in that traditional payment approaches rely on a centralized entity (e.g. the bank) to deal with settlement. This centralized entity may accidentally leak account information, which may be exploited by adversaries to infer sensitive information [11], [16].

2.2 Blockchain-based Distributed Schemes

Recent years have witnessed the rapid developments of blockchain-based distributed schemes to protect privacy. BlockPriv [17] is a location information protection solution designed for the Internet of Things. It considers the spatiotemporal correlation of continuous transactions and realizes location information protection through obfuscation. Li et al. proposed a scheme [18] applied in vehicular ad-hoc networks. They used blockchain to store the hash values of vehicle messages, and hide identity information with K-anonymity unity and dynamic threshold encryption. Shen et al. [19] designed a lightweight threshold certificate authority scheme (LTCA) and a privacy-preserving location-based service protocol (PPVC). LTCA mainly prevents the privacy leak caused by the traditional single-CA online key distribution while PPVC protects privacy by continuously updating the addresses of vehicles on the blockchain.

Compared with the traditional centralized schemes, blockchain-based distributed ones eliminate the single-point-of-failure problem. However, these schemes cannot protect the privacy in payment processing for location-based services. This problem is highly challenging due to the following reasons:

- The payment amount must be accurately calculated based on trusted location information, e.g. entry/exit points, and this unavoidably discloses location information of the vehicles.
- A payment transaction is recorded on a transparent blockchain, and blockchain validators as well as others can access the transaction details, resulting in privacy leak.

Therefore, it is crucial to support correct payments without leaking location information over blockchain. In this paper, we present Vehicloak based on zk-GSigproof, which can effectively address the above challenges.

3 The Vehicloak Model and Preliminaries

In this section, we first describe the major entities in the Vehicloak system model, then define an appropriate threat model, and finally provide preliminaries on group signature, zk-SNARK and BlockMaze.

3.1 System Model

Vehicloak involves four entities: vehicle (V), station (S), toll booth (TB), and blockchain miner (M). S can be an entry station (S_{en}) or an exit station (S_{ex}) according to the actual application scenario.

In this system, V has a pseudonym account V′, which is responsible for interacting with others and uploading public information, such as group signatures, payable amounts and zk-SNARK proofs, to the blockchain. It is assumed that each vehicle has been equipped with a wireless communication technology, e.g. Dedicated Short Range Communication (DSRC), to support communications with the road side units (e.g. the stations).

All stations form a group G and their main task is to generate group signatures for passing vehicles. There can be thousands of stations to provide authenticated entry/exit information, while only one toll booth TB is required to collect toll payments.

The mentioned TB is essentially represented by an account in the blockchain. All payable amounts are eventually transferred to the TB account. It is worth mentioning that TB is not responsible for verifying the correctness of a payable amount, and this task is done by miners.

Miners are responsible for managing the blockchain with a secure consensus algorithm (e.g. proof of work, proof of stake). Our scheme can be compared to a specific application of the Ethereum main chain, where miners can get rewards for their work. Specifically, miners are required to execute a smart contract to determine the legitimacy of the uploaded information, such as group signatures, payable amounts and zk-SNARK proofs.

3.2 Threat Model

In order to capture various attacks, we define the following threat model.

- Vehicle V. We assume that vehicles can be arbitrarily malicious, and act in their best interests. Furthermore, vehicles may collude with each other to maximize their benefits.
- Station S. We assume that S is honest but curious. That is, S would honestly follow the deployed protocols, but it is also interested in inferring the privacy of V, e.g. identities and location information.
- Miner M. Multiple miners follow a secure consensus algorithm to maintain the blockchain. Adversaries cannot compromise the majority of miners to bring down the overall blockchain system.
- Toll booth TB. We assume TB is honest but curious. TB would honestly follow the protocol in payment charging, but it is also interested in inferring the relationship between a vehicle V and a route (inferring the possible route of V). In addition, TB may collude with stations to obtain the private location information of V.

Based on the above threat model, Vehicloak intends to achieve the following privacy and security goals:

- **Location privacy.** Location privacy specifically refers to the passing stations of a vehicle. Assuming that a vehicle V passes through stations S_{en} and S_{ex}. Any adversary should not be able to obtain the correspondence between S_{en} and S_{ex}, avoiding further revealing the corresponding relationship among V, S_{en} and S_{ex}.
- **Identity privacy.** Identity privacy refers to the information of a vehicle, e.g. the identity ID and account address. In Vehicloak, the identities of vehicles should be hidden from the public. No other entity, especially a station, is able to obtain the real identity of an interacting vehicle.
- **Location authenticity.** It is an important task to prove the authenticity of the location information (i.e. the entry/exit stations), because the actual location is one of the necessary conditions for calculating the correct payable amount. The difficulty of this task is to resolve the contradiction between proving the location authenticity to others and protecting the location privacy.
- **Payment correctness.** The payable amount is determined by the information sent to the blockchain by vehicles. Ensuring the correctness of a payment is one of the prerequisites for this scheme to be feasible.

In order to achieve the above security goals, we next introduce the adopted key technologies.

3.3 Group Signature

Group signature, proposed by Chaum et al. [20], can hide the identities of signers in a group. Specifically, each member u can use a unique private key $\mathsf{gsk_u}$ to sign on behalf of the group G. A group signature scheme is composed of five algorithms: Setup (Setup), member join (Join), group signature generation (Sign), group signature verification (Verify), and group signature open (Open). The whole process can be represented by a tuple of polynomial-time algorithms $\Sigma \stackrel{\text{def}}{=}$ (Setup, Join, Sign, Verify, Open):

- $(\mathsf{gpk}, \mathsf{gsk}) \leftarrow \mathsf{Setup}(1^\lambda)$. The Setup algorithm takes a security parameter 1^λ as input, and generates the group public key gpk and the group private key gsk.
- $\mathsf{gsk_u} \leftarrow \mathsf{Join}(\mathsf{gsk}, \mathsf{ID_u})$. When a new member u joins G, the Join algorithm generates a unique private key $\mathsf{gsk_u}$ with gsk and the ID of the new member $\mathsf{ID_u}$.
- $\sigma \leftarrow \mathsf{Sign}(\mathsf{gpk}, \mathsf{gsk_u}, \mathsf{m})$. The Sign algorithm takes the group public key gpk, the private key $\mathsf{gsk_u}$ of u and the message m as inputs to generate the group signature σ for m – σ is the signature of m signed by u on behalf of the group.
- $\{0, 1\} \leftarrow \mathsf{Verify}(\mathsf{gpk}, \mathsf{m}, \sigma)$. The Verify algorithm takes the group public key gpk, the message m and the group signature σ as inputs, and outputs 1 if the verification is successful; otherwise, it outputs 0.
- $\mathsf{gsk_u} \leftarrow \mathsf{Open}(\mathsf{gsk}, \mathsf{m}, \sigma)$. This algorithm is used to restore a member's private key $\mathsf{gsk_u}$ based on the group private key gsk, the message m and the group signature σ when necessary.

During initialization of a group G, the Σ.Setup algorithm relies on a trusted third party (e.g. Certificate Authorities) for key management. After that, according to the requirements of Σ, each G has a trusted manager responsible for

managing and supervising the group members based on the algorithms Σ.Join and Σ.Open. The group members can execute Σ.Sign to generate group signatures and everyone can verify them using Σ.Verify.

3.4 Zero-knowledge Proof and zk-SNARK

Zero-knowledge proofs can prove the correctness of a statement without leaking any additional information [21]. As a type of zero-knowledge proof (computationally sound proof), zero knowledge Succinct Non-interactive ARgument of Knowledge (zk-SNARK) has found its great value in preserving privacy for blockchains such as Zerocash [22] and BlockMaze [4]. This is mainly due to its security features in the succinct proof. zk-SNARK requires a trusted setup phase to generate a common-reference string (crs) as the public parameter, which is used for computing and verifying proofs.

A zk-SNARK scheme can be represented by a tuple of polynomial-time algorithms $\Pi \stackrel{\text{def}}{=}$ (Setup, Prove, Verify). Every statement to be proved with zk-SNARK should be transformed into a circuit C, and it works as follows:

- $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, C)$. The Setup algorithm takes a security parameter 1^λ and a circuit C as inputs to obtain the proving key pk and verification key vk, which constitute the common reference string crs.
- $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, \mathsf{x}, \mathsf{w})$. This algorithm takes the common reference string crs, the circuit public input x and the circuit private input w as inputs, and generates a succinct proof π whose size is irrelevant to the circuit size.
- $\{0,1\} \leftarrow \mathsf{Verify}(\mathsf{crs}, \pi, \mathsf{x})$. This algorithm takes the common reference string crs, a succinct proof π and the circuit public input x as inputs and outputs the verification result: it outputs 1 if verification is successful; and otherwise it outputs 0.

In the algorithm Π.Setup, crs is public, which means that anyone with circuit C can generate and verify a proof. The size of each proof is fixed, with good succinctness. In addition, zk-SNARK also satisfies completeness, computational soundness, computational zero knowledge and non-interactivity [23].

3.5 Privacy-preserving Blockchains

In this paper, we choose BlockMaze, an efficient privacy-preserving blockchain as the underlying blockchain system. BlockMaze adds the zero-knowledge balance to the original Ethereum that takes only plaintext balance. In BlockMaze, each user account has a key pair $(\widetilde{\mathsf{pk}}_u, \widetilde{\mathsf{sk}}_u)$ used for sending transactions. Theoretically, other privacy-preserving blockchains can also be adopted by our scheme.

The zero-knowledge balance zk_balance of user A is defined as follows:

$$\mathsf{zk\_balance} = (\mathsf{addr_A}, \mathsf{value_A}, \mathsf{sn_A}, \mathsf{N_A}, \mathsf{cmt_A}).$$

where $\mathsf{addr_A}$ is A's account address, $\mathsf{value_A}$ is A's account balance, $\mathsf{sn_A}$ is a serial number, $\mathsf{N_A}$ is a random number, and $\mathsf{cmt_A}$ is a commitment over $(\mathsf{value_A}, \mathsf{sn_A}, \mathsf{N_A})$. During the transfer process between user A and B, $\mathsf{cmt_v}$ is a commitment over $(\widetilde{\mathsf{pk}}_B, \mathsf{addr_A}, \mathsf{v}, \mathsf{sn_A}, \mathsf{N_v})$, which is used to update zk_balance, and v represents the transfer amount.

The BlockMaze scheme is composed of 4 polynomial-time algorithms: Mint, Send, Deposit and Redeem. A privacy-preserving transaction between A and B proceeds as follows:

1) A runs the Mint algorithm to generate a transaction named $\mathsf{Tx_{mint}}$, converting A's plaintext balance into zk_balance.
2) A runs the Send algorithm to generate a transaction recorded as $\mathsf{Tx_{send}}$ and a zero-knowledge fund $\mathsf{cmt_v}$ as a commitment to the blockchain. The fund $\mathsf{cmt_v}$ is intended for B.
3) B selects multiple zero-knowledge funds as well as $\mathsf{cmt_v}$ on the blockchain to form a Merkle tree, generating a Merkle proof to prove the corresponding $\mathsf{cmt_v}$ is on the tree. Then B runs the Deposit algorithm to generate $\mathsf{Tx_{deposit}}$ containing the Merkle proof, which transfers the corresponding fund to B's zk_balance. Therefore, the cost of judging whether a $\mathsf{cmt_v}$ is on the tree is low, which effectively alleviates the additional resource consumption caused by privacy protection.
4) Optionally, B can run the Redeem algorithm to generate $\mathsf{Tx_{redeem}}$, transferring its zk_balance to plaintext balance.

Note that the Merkle proof is used to prove whether the value of a specific node is in the set without knowing the values of other nodes. Its storage overhead is small too. Therefore, the cost of judging whether a $\mathsf{cmt_v}$ is on the tree is low, which effectively alleviates the additional resource consumption caused by privacy protection.

Through the above process, BlockMaze can effectively hide the transfer relationship between A and B. No other party can figure out how much money has been sent or received by the two involved parties. However, because smart contracts cannot support the transfer of zero-knowledge funds, BlockMaze can not be applied to transfer operations between smart contracts and users. Fortunately, one can use the pseudonym technique to solve the problem mentioned above, as detailed in Sec. 5.2.

4 zk-GSigproof: a zk-SNARK with Group Signature

In this section, we propose zk-GSigproof, a special zk-SNARK that makes use of the advantages of group signatures. It provides basic technical support for Vehicloak and is proven to satisfy data authenticity and authenticator anonymity in addition to the standard properties of zk-SNARKs.

4.1 Introduction of zk-GSigproof

In Vehicloak, we aim to enable a vehicle to securely upload payment information and correctly complete the payment operation, while eliminating the reliance on a third-party billing agency. However, under the assumption that the vehicle is arbitrarily malicious, how to ensure the correctness and privacy of payment information becomes difficult.

Although the current techniques such as zero-knowledge proofs can help realize privacy-preserving computing, they usually cannot guarantee the authenticity of the input information; while the general signature techniques can prove the authenticity of information but expose the identity privacy of the signer. These observations motivate us to propose zk-GSigproof, which combines group signature and zero-knowledge proof to ensure the authenticity of the uploaded information and the correctness of the amount to be paid while eliminating the risk of privacy leak during information uploading and payment calculation, since group signature can hide the address of the signer, perfectly making up for the deficiencies of zero-knowledge proof mechanisms.

In the following, we give the definition of zk-GSigproof.

Definition 1 (zk-GSigproof). zk-GSigproof *is a zk-SNARK for arithmetic circuit, and is composed of 4 polynomial-time algorithms* $\Delta \stackrel{def}{=}$ (Setup, DataAuth, Prove, Verify).

We claim that zk-GSigproof satisfies data authenticity and authenticator anonymity, in addition to standard zk-SNARK properties, i.e. perfect completeness, succinctness, knowledge extraction and computational zero-knowledge. The construction of zk-GSigproof Δ is based on a zk-SNARK algorithm Π and a group signature scheme Σ. It works as follows:

$(\mathsf{crs}, \mathsf{gpk}, \mathsf{gsk_u}) \leftarrow \mathsf{Setup}(1^\lambda, C)$. On inputs of a security parameter λ and a circuit C, the algorithm first runs $\Sigma.\mathsf{Setup}(1^\lambda)$ to generate the group public key gpk and group private key $\mathsf{gsk_u}$ for group member $\mathsf{ID_u}$, then runs $\Pi.\mathsf{Setup}(1^\lambda, C)$ to produce a common reference string crs, and finally publishes crs and gpk.

Note that crs is related to the structure of C (like the number of inputs and their operations) and is independent of the specific input assignment. The specific circuit logic used in Vehicloak is shown in Fig. 1. The private input w includes entry/exit stations ($\mathsf{S_{en}}/\mathsf{S_{ex}}$), two random numbers ($\mathsf{N_{en}}, \mathsf{N_{ex}}$) and the pre-deposit identification $\mathsf{Tx_{id}}$ (details shown in Sec. 5.2.2). The calculation result F represents the amount payable and the hash results ($\mathsf{H_{en}}$ and $\mathsf{H_{ex}}$) are to be signed with the algorithm Σ.Sign.

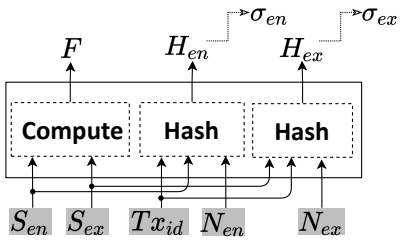

Fig. 1. The logic diagram of the circuit in zk-GSigproof. The inputs with gray background are private ones protected with zk-SNARK proof.

$\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, \mathsf{x}, \mathsf{w})$. This algorithm generates a proof π based on $\Pi.\mathsf{Prove}(\mathsf{crs}, \mathsf{x}, \mathsf{w})$. Users can generate their own proofs with the same crs and different input assignment.

$\sigma \leftarrow \mathsf{DataAuth}(\mathsf{gpk}, \mathsf{gsk_u}, \mathsf{m})$. For each message m, such as the hash result $\mathsf{H_{en}}$ or $\mathsf{H_{ex}}$, this algorithm runs $\Sigma.\mathsf{Sign}(\mathsf{gpk}, \mathsf{gsk_u}, \mathsf{m})$ and obtains a group signature σ.

$\{0,1\} \leftarrow \mathsf{Verify}(\mathsf{crs}, \mathsf{gpk}, \pi, \mathsf{x}, \mathsf{m}, \sigma)$. This algorithm first runs $\Pi.\mathsf{Verify}(\mathsf{crs}, \pi, \mathsf{x})$ to verify the proof π with the public input x and the common reference string crs. Then, it runs $\Sigma.\mathsf{Verify}(\mathsf{gpk}, \mathsf{m}, \sigma)$ to verify the group signature with the group public key gpk and the message m. It outputs 1 if the proof π and the group signature σ are valid and 0 otherwise.

The additional properties of the zk-GSigproof are defined as follows.

- **Data Authenticity**. The prover can convince the verifier that the output F is computed from the private input w authenticated by group signatures $\sigma_i|_{i\in T}$, where T is a given set containing multiple instances. Formally, we define the following experiment:

$\mathbf{Exp}^{\mathsf{auth}}_{\mathsf{zk\text{-}GSigproof},\mathcal{A}}(1^\lambda, C)$:
$(\mathsf{crs}, \mathsf{gpk}, \mathsf{gsk_u}) \leftarrow \mathsf{Setup}(1^\lambda, C)$
$(\pi, \mathsf{x}, \sigma_i|_{i\in\mathsf{T}}) \leftarrow \mathcal{A}^{\mathsf{DataAuth}(\mathsf{gsk_u},\cdot)}(\mathsf{crs})$
$\mathsf{w} \leftarrow \mathcal{E}(\mathsf{trans}_\mathcal{A})$
if $((\mathsf{x}, \mathsf{w}), \sigma_i) \notin \mathcal{S}$ and
  $\mathsf{Verify}(\mathsf{crs}, \mathsf{gpk}, \pi, \mathsf{x}, \mathsf{m_i}|_{i\in\mathsf{T}}, \sigma_i|_{i\in\mathsf{T}}) = 1$
  return 1
else return 0

Here $\mathcal{A}$ is a non-uniform polynomial-time adversary, $\mathcal{S}$ is the list including all messages signed with legitimate gsk_u, $\mathsf{trans}_\mathcal{A}$ is the list containing all $\mathcal{A}$'s inputs, outputs and randomness. This experiment represents the probability that the adversary composes a valid instance with a zero knowledge proof without querying the signing oracle.
We say that a zk-GSigproof scheme achieves data authenticity if for any non-uniform polynomial-time adversary $\mathcal{A}$, there exists a probabilistic polynomial-time witness extractor $\mathcal{E}$ such that the probability $\Pr[\mathbf{Exp}^{\mathsf{auth}}_{\mathsf{zk\text{-}GSigproof},\mathcal{A}}(1^\lambda, C) = 1]$ is negligible.

- **Authenticator Anonymity**. This property is defined as the notion that an adversary cannot link a zk-GSigproof to the corresponding authenticator. More formally, we define the following experiment:

$\mathbf{Exp}^{\mathsf{anon}}_{\mathsf{zk\text{-}GSigproof},\mathcal{A}}(1^\lambda, C)$:
$(\mathsf{crs}, \mathsf{gpk}, \mathsf{gsk_u}) \leftarrow \mathbf{Setup}(1^\lambda, C)$
$(\mathsf{u}_0, \mathsf{u}_1, \mathsf{m}, \mathsf{state}) \leftarrow$
  $\mathcal{A}_1^{\mathsf{Setup}(\cdot),\mathsf{Prove}(\cdot),\mathsf{Verify}(\cdot)}(\mathsf{crs}, \mathsf{gpk})$
$b \stackrel{\$}{\leftarrow} \{0,1\}$
$\sigma^* \leftarrow \mathsf{DataAuth}(\mathsf{gpk}, \mathsf{gsk_{u_b}}, \mathsf{m})$
$b' \leftarrow \mathcal{A}_2^{\mathsf{Setup}(\cdot),\mathsf{Prove}(\cdot),\mathsf{Verify}(\cdot)}(\sigma^*, \mathsf{state})$
if $b' = b$ return 1
else return 0

It represents the probability that the adversary composes a valid instance with a zero knowledge proof without querying the signing oracle.
We say that a zk-GSigproof scheme achieves authenticator anonymity if for any non-uniform

polynomial-time adversary $\mathcal{A}$, the probability $\Pr[\mathbf{Exp}^{\mathsf{anon}}_{\mathsf{zk\text{-}GSigproof},\mathcal{A}}(1^\lambda, C) = 1]$ is negligible.

For the security of the above construction, we have the following theorem.

Theorem 1. *If* Π *is a zk-SNARK scheme satisfying completeness, soundness, succinctness and zero knowledge, and* Σ *is a secure group signature scheme, then the above construction of* zk-GSigproof *satisfies data authenticity and authenticator anonymity in addition to completeness, soundness, succinctness and zero knowledge.*

The proof of the above theorem is deferred to Sec. 4.2.

4.2 The Security Proof of zk-GSigproof

Proof. It is trivial to prove that the zk-GSigproof construction satisfies completeness and succinctness. The soundness of our construction can also be easily reduced to the soundness of the corresponding zk-SNARK scheme. In the following, we prove data authenticity of our zk-GSigproof construction.

As described in Sec. 4.1, we need to show that the probability $\Pr[\mathbf{Exp}^{\mathsf{auth}}_{\mathsf{zk\text{-}GSigproof},\mathcal{A}}(1^\lambda, C) = 1]$ is negligible. Due to the soundness of our zk-GSigproof construction, it implies that the probability of the following experiment returning 1 is negligible:

$\mathbf{Exp}^{\mathsf{sound}}_{\mathsf{zk\text{-}GSigproof},\mathcal{A}}(1^\lambda, C)$:
  $(\mathsf{crs}, \mathsf{gpk}, \mathsf{gsk_u}) \leftarrow \mathsf{Setup}(1^\lambda, C)$
  $(\pi, \mathsf{x}, \sigma_\mathsf{i}|_{\mathsf{i}\in\mathsf{T}}) \leftarrow \mathcal{A}^{\mathsf{DataAuth}(\mathsf{gsk}_u,\cdot)}(\mathsf{crs})$
  $\mathsf{w} \leftarrow \mathcal{E}(\mathsf{trans}_\mathcal{A})$
  if $(\mathsf{x}, \mathsf{w}) \notin \mathcal{R}_C$ and
    $\mathsf{Verify}(\mathsf{crs}, \mathsf{gpk}, \pi, \mathsf{x}, \mathsf{m_i}|_{\mathsf{i}\in\mathsf{T}}, \sigma_\mathsf{i}|_{\mathsf{i}\in\mathsf{T}}) = 1$
    return 1
  else return 0

Here $\mathcal{R}_C$ is the relation defined by the circuit C.

Denote $\mathbf{E}_1$ as the event that $\mathsf{Verify}(\mathsf{crs}, \mathsf{gpk}, \pi, \mathsf{x}, \mathsf{m_i}|_{\mathsf{i}\in\mathsf{T}}, \sigma_\mathsf{i}|_{\mathsf{i}\in\mathsf{T}}) = 1$, $\mathbf{E}_2$ as the event that $(\mathsf{x}, \mathsf{w}) \notin \mathcal{R}_C$, and $\mathbf{E}_3$ as the event $((\mathsf{x}, \mathsf{w}), \sigma_\mathsf{i}) \notin \mathcal{S}$. Then $\Pr[\mathbf{Exp}^{\mathsf{sound}}_{\mathsf{zk\text{-}GSigproof},\mathcal{A}}(1^\lambda, C) = 1] = \Pr[\mathbf{E}_1 \cap \mathbf{E}_2]$ and $\Pr[\mathbf{Exp}^{\mathsf{auth}}_{\mathsf{zk\text{-}GSigproof},\mathcal{A}}(1^\lambda, C) = 1] = \Pr[\mathbf{E}_1 \cap \mathbf{E}_3]$. So we have

$$\begin{aligned}
&\Pr[\mathbf{Exp}^{\mathsf{auth}}_{\mathsf{zk\text{-}GSigproof},\mathcal{A}}(1^\lambda, C) = 1] = \Pr[\mathbf{E}_1 \cap \mathbf{E}_3] \\
&= \Pr[\mathbf{E}_1 \cap \mathbf{E}_3 \cap \bar{\mathbf{E}_2}] + \Pr[\mathbf{E}_1 \cap \mathbf{E}_3 \cap \mathbf{E}_2] \\
&\leq \Pr[\mathbf{E}_3 \cap \bar{\mathbf{E}_2} | \mathbf{E}_1] \cdot \Pr[\mathbf{E}_1] + \Pr[\mathbf{E}_1 \cap \mathbf{E}_2] \\
&\leq \mathsf{Adv}^{\mathsf{hash}}_{\mathcal{A}}(1^\lambda) + \Pr[\mathbf{Exp}^{\mathsf{sound}}_{\mathsf{zk\text{-}GSigproof},\mathcal{A}}(1^\lambda, C) = 1]
\end{aligned}$$

Here $\mathsf{Adv}^{\mathsf{hash}}_{\mathcal{A}}(1^\lambda)$ denotes the advantage of the adversary breaking the hash function. Note that event $\mathbf{E}_1$ represents validity of σ_i with regard to H, and event $\bar{\mathbf{E}_2}$ represents H is the hash of w. Therefore, event $\mathbf{E}_3 \cap \bar{\mathbf{E}_2} | \mathbf{E}_1$ happens only if the adversary can find w' such that $\mathsf{H} = \mathsf{hash}(\mathsf{x}, \mathsf{w}')$, where $((\mathsf{x}, \mathsf{w}'), \sigma_\mathsf{i}) \notin \mathcal{S}$.

Since both $\mathsf{Adv}^{\mathsf{hash}}_{\mathcal{A}}(1^\lambda)$ and $\Pr[\mathbf{Exp}^{\mathsf{sound}}_{\mathsf{zk\text{-}GSigproof},\mathcal{A}}(1^\lambda, C) = 1]$ are negligible, $\Pr[\mathbf{Exp}^{\mathsf{auth}}_{\mathsf{zk\text{-}GSigproof},\mathcal{A}}(1^\lambda, C) = 1]$ must be negligible.

The authenticator anonymity of zk-GSigproof can be reduced to the full anonymity of the underlying group signature scheme as in [24].

To sum up, our zk-GSigproof construction satisfies data authenticity, authenticator anonymity in addition to the standard properties of zk-SNARK. □

5 VEHICLOAK: A ZK-GSIGPROOF ENABLED DECENTRALIZED PRIVACY-PRESERVING PAYMENT SCHEME

In this section, we first provide an overview on Vehicloak, then explain the protocol in detail, and finally discuss a few practical considerations and analyze a number of privacy and security problems of Vehicloak.

5.1 Overview

Assuming that the Vehicloak has been successfully initialized (**Initialize**), vehicles act as blockchain users to send payment transactions, and all stations form a group to generate group signatures.

Suppose a vehicle V joins Vehicloak and its route is from $\mathsf{S_{en}}$ to $\mathsf{S_{ex}}$. Before V enters $\mathsf{S_{en}}$, it first calls the smart contract (Contract-Vehicloak) to prepay amount F′, which must be greater than the maximum amount $\mathsf{F_{max}}$. We define the above process as **Pre-deposit**, and each pre-deposit transaction ultimately corresponds to a unique $\mathsf{Tx_{id}}$, which is used as a certificate for V to enter $\mathsf{S_{en}}$.

After V enters $\mathsf{S_{en}}$, $\mathsf{S_{en}}$ first computes the hash value $\mathsf{H_{en}}$ of its address, then generates the group signature σ_{en} for $\mathsf{H_{en}}$, and finally sends both $\mathsf{H_{en}}$ and σ_{en} to V. By the same way, V obtains the hash value $\mathsf{H_{ex}}$ and the group signature σ_{ex} from $\mathsf{S_{ex}}$. After leaving $\mathsf{S_{ex}}$, V calls Contract-Vehicloak to pay the payable amount F. In order to prove correctness of F while preserving location privacy, V needs to generate a zero-knowledge proof, which requires the corresponding circuit shown in Fig. 1. Finally, V uploads the payable amount F, the zk-SNARK proof, and the group signatures of entry/exit to the blockchain. This is the **Upload** process of Vehicloak.

In **Validate**, the miners are responsible for verifying the on-chain information to obtain rewards (just like the main chain of Ethereum). We add the verification process to Contract-Vehicloak, so that miners only need to execute Contract-Vehicloak to prove the correctness of the prepay amount F′, the payable amount F, the zk-SNARK proof, and the group signatures of entry/exit.

The **Settlement** process is automatically executed by Contract-Vehicloak. When **Validate** is successful, the amount equivalent to F is transferred to the TB account, and the remaining amount (F′ - F) is returned to V's account. If the verification fails or malicious behavior exists, e.g. timeout, the prepay amount F′ is deducted for punishment.

We summarize the entire process in Fig. 2, and describe the smart contract and basic protocol in Fig. 3 and Fig. 4, respectively.

5.2 The Basic Protocol

In this section, we first detail the entire protocol, i.e., the 5 steps of Vehicloak: **Initialize**, **Pre-deposit**, **Upload**, **Validate** and **Settlement**. Then, we add a **Timeout** mechanism to punish vehicles for malicious behaviors.

5.2.1 Initialize

During the initialization process, vehicle V registers on Vehicloak to get its pseudonym V′ and obtain the key pair $(\widetilde{\mathsf{pk}}_{V'}, \widetilde{\mathsf{sk}}_{V'})$. Note that V has two accounts in blockchain,

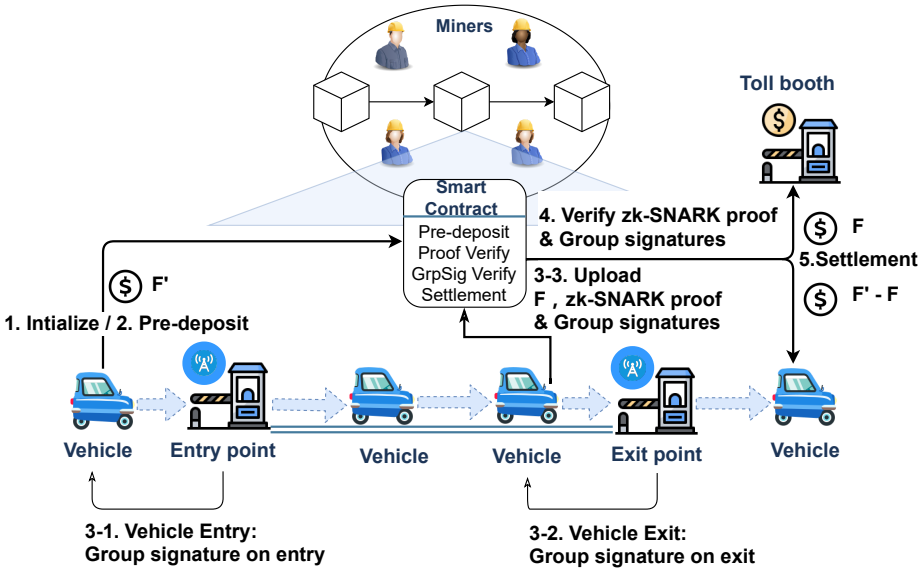

Fig. 2. Vehicloak contains 5 steps marked from 1 to 5. Step 1 represents **Initialize**, which is responsible for initializing the system. Step 2 refers to **Pre-deposit** and transfers the prepay amount to the smart contract. Step 3 describes the **Upload** process, which is responsible for generating the verifiable information and uploading them to blockchain. Step 4 illustrates the **Validate** process, in which miners execute the smart contract to verify the uploaded information. According to the verification result, the amount is transferred to the vehicle and the toll booth in Step 5 **Settlement**.

Contract-Vehicloak

Initialize:
1) Set state := INIT.
Pre-deposit:
Upon receiving ("Pre-deposit", F′) from pseudonym V′:
1) Assert state = INIT;
2) Assert F′ > amount threshold F_{max};
3) Set state = DEPOSIT.
Validate:
Upon receiving ("Upload", π, σ_{en}, σ_{ex}, H_{en}, H_{ex}, F) from V′:
1) Assert state = DEPOSIT;
2) Set state = UPLOAD;
3) Verify (π, σ_{en}, σ_{ex}) based on zk-GSigproof Δ:
If Δ.Verify(crs, gpk, π, F, H_{en}, H_{ex}, σ_{en}, σ_{ex}) = 1:
Set state = VALIDATE-TRUE;
Settlement.
Else:
Set state = VALIDATE-FALSE;
Settlement.
Settlement:
1) If state = VALIDATE-TRUE:
Transfer F ⇒ TB;
Transfer (F′-F) ⇒ V′;
2) Else:
Transfer F′ ⇒ TB.
3) Set state = FINISHED.
Timeout:
When state = DEPOSIT, Timer T starts timing:
1) If current T > T_{end} and state ≠ UPLOAD:
Transfer F′ ⇒ TB.
2) Set state = TIMEOUT.

Fig. 3. The smart contract of Vehicloak (Contract-Vehicloak).

with one corresponding to its true identity V and one to the pseudonym V'. The pseudonym account is used only once for the particular trip while the true account stays in blockchain as needed. V also receives the common reference string crs from the group that consists of all stations. The crs is used to generate/verify proofs for the system. Each station S obtains its key pair $(\mathsf{gpk}, \mathsf{gsk}_S)$ based on the Setup algorithm in zk-GSigproof (Δ), where gsk_S is used to sign on behalf of the entire group G.

$$(\mathsf{crs}, \mathsf{gpk}, \mathsf{gsk}_S) = \Delta.\mathsf{Setup}(1^\lambda, C).$$

5.2.2 Pre-deposit

V transfers the prepay amount F′ to its pseudonym account V′. We emphasize that a privacy-preserving cryptocurrency must be used to protect the identity privacy by hiding the transfer relationship between two parties, so we resort to BlockMaze (shown in 3.5). First, V executes the BlockMaze.Mint algorithm to convert its deposit prepay amount F′ to cmt_v and update its zero-knowledge balance zk_balance. V runs the BlockMaze.Send algorithm to send cmt_v to the blockchain. Then, the pseudonym V′ builds a Merkle tree, whose leaf nodes are based on multiple cmt_v in the blockchain network. V′ generates a Merkle proof based on this tree, executes BlockMaze.Deposit to prove that its cmt_v exists on the tree and get cmt_v. After that, V′ runs BlockMaze.Redeem to update its zero-knowledge balance zk_balance and convert zk_balance into plaintext balance. Finally, V′ sends $\mathsf{Tx}_{\text{Pre-deposit}}$ to transfer F′ to Contract-Vehicloak. We set the address of $\mathsf{Tx}_{\text{Pre-deposit}}$ as the identification of **Pre-deposit**, recorded as $\mathsf{Tx}_{\mathsf{id}}$.

$$\mathsf{Tx}_{\text{Pre-deposit}} \stackrel{\text{def}}{=} (\mathsf{From} : V', \mathsf{To} : \mathsf{contract}, \mathsf{Amount} : F').$$

The Vehicloak Protocol Details

Vehicle V with its pseudonym V′:

Initialize:
1) Get $(\widetilde{\mathsf{pk}}_{V'}, \widetilde{\mathsf{sk}}_{V'})$ from blockchain;
2) Get $\mathsf{crs} = \mathsf{zk\text{-}GSigproof}\ \Delta.\mathsf{Setup}(1^\lambda, C)$.

Pre-deposit:
Upon arriving the entry of station $\mathsf{S_{en}}$:
1) Send ("Pre-deposit", F′) → contract;
2) Store ("Pre-deposit", F′) as $\mathsf{Tx_{id}}$;
3) Send $\mathsf{Tx_{id}} \rightarrow \mathsf{S_{en}}$.

Upload:
Upon receiving (σ_{en}, $\mathsf{H_{en}}$, $\mathsf{SN_{en}}$) and (σ_{ex}, $\mathsf{H_{ex}}$, $\mathsf{SN_{ex}}$) from [$\mathsf{S_{en}}$, $\mathsf{S_{ex}}$]:
1) Get random numbers ($\mathsf{N_{en}}$, $\mathsf{N_{ex}}$) = $\mathsf{ECC}(\widetilde{\mathsf{sk}}_{V'}, \mathsf{SN_{en}}, \mathsf{SN_{ex}})$;
2) Generate proof $\pi = \Delta.\mathsf{Prove}(\mathsf{crs}, \mathsf{S_{en}}, \mathsf{S_{ex}}, \mathsf{N_{en}}, \mathsf{N_{ex}}, \mathsf{Tx_{id}})$;
3) Send ("Upload", π, σ_{en}, σ_{ex}, $\mathsf{H_{en}}$, $\mathsf{H_{ex}}$, F) → contract.

Station S [Entry station $\mathsf{S_{en}}$, Exit station $\mathsf{S_{ex}}$]:

Initialize:
1) Get ($\mathsf{gpk}, \mathsf{gsk_S}) = \Delta.\mathsf{Setup}(1^\lambda, C)$.

Upload:
[$\mathsf{S_{en}}$]: Upon receiving $\mathsf{Tx_{id}}$ from V′:
1) Assert the legitimacy of $\mathsf{Tx_{id}}$;
2) Generate a new random 16-bit integer $\mathsf{N_{en}}$;
3) Compute $\mathsf{H_{en}} = \mathsf{Hash}(\mathsf{S_{en}}, \mathsf{Tx_{id}}, \mathsf{N_{en}})$;
4) Get signature $\sigma_{en} = \Delta.\mathsf{DataAuth}(\mathsf{gpk}, \mathsf{gsk_S}, \mathsf{H_{en}})$;
5) Encrypt $\mathsf{SN_{en}} = \mathsf{ECC}(\widetilde{\mathsf{pk}}_{V'}, \mathsf{N_{en}})$;
6) Send (σ_{en}, $\mathsf{H_{en}}$, $\mathsf{SN_{en}}$) → V′.

[$\mathsf{S_{ex}}$]: V′ is about to leaving:
7) Generate a new random 16-bit integer $\mathsf{N_{ex}}$;
8) Compute $\mathsf{H_{ex}} = \mathsf{Hash}(\mathsf{S_{ex}}, \mathsf{Tx_{id}}, \mathsf{N_{ex}})$;
9) Get signature $\sigma_{ex} = \Delta.\mathsf{DataAuth}(\mathsf{gpk}, \mathsf{gsk_S}, \mathsf{H_{ex}})$;
10) Encrypt $\mathsf{SN_{ex}} = \mathsf{ECC}(\widetilde{\mathsf{pk}}_{V'}, \mathsf{N_{ex}})$;
11) Send (σ_{ex}, $\mathsf{H_{ex}}$, $\mathsf{SN_{ex}}$) → V′.

Fig. 4. The Vehicloak protocol details.

5.2.3 Upload

When V arrives at entry station $\mathsf{S_{en}}$, $\mathsf{S_{en}}$ calculates a hash value $\mathsf{H_{en}}$ of $\mathsf{N_{en}}$, $\mathsf{S_{en}}$ and $\mathsf{Tx_{id}}$. $\mathsf{N_{en}}$, generated by $\mathsf{S_{en}}$ using the BBS generator [25], is a new random 16-bit integer to enhance the randomness of the hash result. Then, $\mathsf{S_{en}}$ generates a group signature σ_{en} for $\mathsf{H_{en}}$ and sends σ_{en}, $\mathsf{H_{en}}$ and $\mathsf{SN_{en}}$ to V′. $\mathsf{SN_{en}}$ is the encrypted random number based on the public key $\widetilde{\mathsf{pk}}_{V'}$ and the random number $\mathsf{N_{en}}$, and we can use the algorithm based on the Elliptic Curve Cryptography (ECC) [26] to implement that. By the same way, when V arrives at $\mathsf{S_{ex}}$, it also gets the signature result σ_{ex}, the hash value $\mathsf{H_{ex}}$ and the encrypted random number $\mathsf{SN_{ex}}$.

$$\mathsf{H_{en/ex}} = \mathsf{Hash}(\mathsf{N_{en/ex}}, \mathsf{Tx_{id}}, \mathsf{S_{en/ex}}),$$

$$\sigma_{en/ex} = \Delta.\mathsf{DataAuth}(\mathsf{gpk}, \mathsf{gsk}, \mathsf{H_{en/ex}}).$$

In addition, V′ needs to generate a zero-knowledge proof to hide the location information while ensuring the correctness of the payable amount F. Based on the zk-SNARK protocol, V needs to generate the circuit C according to Fig. 1. The whole circuit includes Compute and Hash functions, where Compute calculates the payable amount F based on the private location information ($\mathsf{S_{en}}$, $\mathsf{S_{ex}}$) and Hash calculates the hash results ($\mathsf{H_{en}}$, $\mathsf{H_{ex}}$) based on $\mathsf{S_{en}}$, $\mathsf{S_{ex}}$, $\mathsf{N_{en}}$, $\mathsf{N_{ex}}$ and $\mathsf{Tx_{id}}$. Then, V′ generates a proof π based on the algorithm $\Delta.\mathsf{Prove}$.

$$\pi = \Delta.\mathsf{Prove}(\mathsf{crs}, \mathsf{S_{en}}, \mathsf{S_{ex}}, \mathsf{N_{en}}, \mathsf{N_{ex}}, \mathsf{Tx_{id}}).$$

Finally, V′ packages F, π, σ_{en}, σ_{ex}, $\mathsf{H_{en}}$ and $\mathsf{H_{ex}}$ into a new transaction $\mathsf{Tx_{Upload}}$ and sends it to blockchain.

$$\mathsf{Tx_{Upload}} \stackrel{\text{def}}{=} (\pi, \sigma_{en}, \sigma_{ex}, \mathsf{H_{en}}, \mathsf{H_{ex}}, \mathsf{F}).$$

5.2.4 Validate

When $\mathsf{Tx_{Upload}}$ is sent successfully, miners execute Contract-Vehicloak for verification. There are three main components of verification: π, σ_{en} and σ_{ex}. The parameters required for the $\Delta.\mathsf{Verify}$ algorithm are common reference string crs, group public key gpk, proof π, group signature σ, payable amount F and hash value H. Among them, crs and gpk are publicly generated by $\Delta.\mathsf{Setup}$, and other parameters are provided by $\mathsf{Tx_{Upload}}$.

$$\{1, 0\} = \Delta.\mathsf{Verify}(\mathsf{crs}, \mathsf{gpk}, \pi, \mathsf{F}, \mathsf{H_{en}}, \mathsf{H_{ex}}, \sigma_{en}, \sigma_{ex}).$$

5.2.5 Settlement

If the verification is successful, Contract-Vehicloak transfers F to the toll booth TB, and returns the remaining amount (F′ - F) to vehicle's pseudonym account V′. Otherwise, V′ is judged to be illegal and the prepay amount F′ is transferred to TB as punishment. The final task is to transfer the balance from V′ to V while not revealing the transfer relationship between them. First, V′ executes BlockMaze.Mint to convert plaintext balance into zk_balance. Then, V′ converts zk_balance to $\mathsf{cmt_v}$ and sends $\mathsf{cmt_v}$ to the blockchain based on BlockMaze.Send. V selects multiple $\mathsf{cmt_v}$ to form a Merkle tree, and executes BlockMaze.Deposit to prove that its $\mathsf{cmt_v}$ exists on this tree and get $\mathsf{cmt_v}$. Finally, V runs BlockMaze.Redeem to convert $\mathsf{cmt_v}$ into plaintext balance.

5.2.6 Timeout

We set a threshold $\mathsf{T_{en}}$ to prompt V′ to complete the payment. Specifically, when V′ completes step **Pre-deposit**, the timer T of Contract-Vehicloak starts timing. When T exceeds $\mathsf{T_{en}}$ and V′ does not complete step **Upload**, Contract-Vehicloak would transfer all the pre-deposited amount to TB as a penalty. On one hand, this timeout mechanism makes it unnecessary for vehicles to pay the amount immediately when they leave, reducing the possibility of congestion during peak traffic hours. On the other hand, it effectively prevents the malicious behavior of vehicles from not uploading information to avoid payment.

5.3 Discussions

5.3.1 Extending to a Generalized Framework

Vehicloak has good universality in real payment scenarios, and its application is not limited to IoV (Internet of Vehicle).

In this subsection, we first discuss a few practical application scenarios of Vehicloak and then abstract a generalized framework.

In the highway toll collection application exemplified in 5.2, the calculation of the payable amount depends on location information. However, for applications such as parking charging, the payable amount depends on time, i.e. the entry time (T_{en}) and the exit time (T_{ex}). A slight modification to the circuit should suffice. Specifically, we add T_{en} and T_{ex} as the inputs of the Compute function to compute the payable amount F, and the time information is also used as part of the hash calculation. For other applications such as calculating electricity bills and shopping checkout, our scheme is also applicable. Taking the electricity billing as an example, we should reduce the number of inputs of the circuit to 1, compute the bill based on the electricity consumption E, and then use E to calculate the hash value for group signature verification.

One may notice that different scenarios affect the logic of the circuit, thus we abstract a generalized one as shown in Fig. 5. The required parameters of the amount computing function Compute are recorded as obj, which are private (e.g. E for the electricity billing, T_{en} and T_{ex} for the parking charging, and S_{en} and S_{ex} for highway toll collection). The number of obj and hash calculations Hash can be arbitrary, recorded as n. Each hash calculation contains a random value N, which is for the sake of enhancing randomness. It is worth noting that the description in Sec. 5.1 is for the case of $n = 2$, which is applicable to scenarios such as highway toll collection and parking charging.

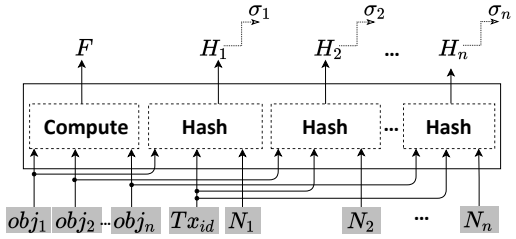

Fig. 5. The generalized logic diagram of the circuit for different payment scenarios. The inputs with gray background are private ones protected by the zk-SNARK proof.

In summary, our scheme is not limited to IoV, it is also applicable to other payment scenarios. However, one may notice that the collection of obj depends on a certain number of entities (e.g. the station S for highway toll collection), but it is reasonable. Currently in practice, there exist one or more central servers (e.g. smart meters, toll booths and electronic cashiers), which are responsible for not only collecting information but also calculating the payable amount. Our scheme changes the role of these servers so that they are only responsible for collecting the payment-related information, which enables our solution to be quickly applied to various real-life scenarios without destroying the original infrastructure.

5.3.2 Alternative Zero-knowledge Proofs and Privacy-preserving Blockchains

In this study, we adopt Groth16 [27] in zk-GSigproof because it is the most optimized zk-SNARK in terms of proof size. However, Groth16 needs to establish a trusted setup, relies on a one-time common reference string, and has "toxic waste". Fortunately, zk-GSigproof has a strong reconfigurability, and can be combined with other zero-knowledge proof schemes such as the transparent setup scheme [28] and the universal setup scheme [29], based on the actual needs. The transparent setup scheme [28] does not rely on any trusted setup; it solves the problem of "toxic waste" and has higher security, however, it has poor succinctness. The universal setup scheme [29] can solve the one-time use of reference string problem, but it is inferior to the transparent setup in terms of algorithm security and its proof size requires greater storage consumption than Groth16.

Additionally, we employ the privacy-preserving blockchain Blockmaze in Vehicloak to cut off the association between an account and its pseudonym, protecting the former's identity and route records. Another reason of adopting Blockmaze is because Blockmaze is implemented based on zk-SANRK, which perfectly fits the need of our zk-GSigproof without extra work. However, although Blockmaze meets our technical needs, it brings considerable changes to Ethereum, making Blockmaze-enabled Vehicloak not directly usable in Ethereum. Fortunately, in fact, the implementation of zk-GSigproof does not depend on Blockmaze. In Vehicloak, Blockmaze and zk-GSigproof jointly provide a secure and private environment, but they are not related to each other. Therefore, one can replace Blockmaze with other privacy-preserving blockchains such as AttriChain [30] and Zether [31] to implement Vehicloak in Ethereum without modifications to the underlying blockchain.

5.3.3 Security and Privacy Analysis

In this subsection, we deeply analyze the security goals proposed in Sec. 3.2.

Location Privacy. Location privacy is well protected because the original location information is never exposed in public. Suppose there is an adversary $\mathcal{A}$ who intends to obtain the location information of the vehicle V. There are two cases: $\mathcal{A}$ would try to intercept the information sent by stations to vehicles, or $\mathcal{A}$ would obtain useful location information from $\mathsf{Tx}_{\mathrm{Upload}}$.

As for the first case, according to zk-GSigproof, the location information is protected by hashing. Therefore, even if $\mathcal{A}$ can obtain this information, it is impossible to determine the specific location of vehicles. As for the second case, zk-GSigproof hides the location information as the private inputs of zk-SNARK, and $\mathcal{A}$ cannot get the private inputs based on the proof and its public inputs.

Identity Privacy. In Vehicloak, we protect vehicles' identity privacy based on a privacy-preserving scheme BlockMaze and pseudonyms. Suppose there is an adversary $\mathcal{A}$ who colludes with a station S_{adv} and tries to obtain the identity information. There are two cases: $\mathcal{A}$ would get the identity of passing vehicles through S_{adv}. Besides that, the transparent transfer relationship on the blockchain is also a hidden danger of identity exposure.

As for the first case, each vehicle would get a pseudonym account for a particular trip. All the interaction processes among vehicles and other entities (e.g. vehicles and stations, vehicles and blockchain) are completed through

pseudonyms, so $\mathcal{A}$ cannot directly obtain the true identity information through $\mathsf{S}_{\mathsf{adv}}$. As for the second case, transparent and traceable blockchain transaction records would expose the direct relationship between a vehicle's real account and its pseudonym account. Vehicloak resorts to BlockMaze, which hides the direct relationship between senders and receivers based on the zero-knowledge balance. Therefore, $\mathcal{A}$ can only obtain the information of the vehicle's pseudonym account and cannot get further information of its real identity.

Location Authenticity and Amount Correctness. Vehicloak can ensure the location authenticity and correctness of the payable amount and effectively resist the over-spending attack based on zk-GSigproof. Suppose there is an adversary $\mathcal{A}$ who intends to illegally prepay more amount or reduce the payable amount. There are two cases: $\mathcal{A}$ would attempt to forge F' in $\mathsf{Tx}_{\text{Pre-deposit}}$ to get more pre-deposit amount (over-spending attack) or $\mathcal{A}$ would provide a fake F or location information to pay less.

However, as for the first case, Vehicloak stipulates that each prepay amount must be greater than $\mathsf{F}_{\mathsf{max}}$. In order to remove human intervention, we use smart contract Contract-Vehicloak (shown in Fig. 3) to realize the verification of the amount. Each $\mathsf{Tx}_{\text{Pre-deposit}}$ is strictly limited to the logic of Contract-Vehicloak and public supervision. As for the second case, zk-GSigproof has been proven to have data authenticity, and the details are shown in Sec. 4.2. It combines the advantages of zero-knowledge proof and group signature to protect location privacy while ensuring the correctness of the payable amount and the authenticity of location.

6 Implementation and Performance Evaluation

In this section, we describe the concrete implementation of Vehicloak and test its performance. Specifically, we first elaborate on the implementation of Vehicloak, then simulate realistic scenarios on Aliyun, and finally summarize the experimental results.

6.1 Implementation

The implementation of Vehicloak mainly includes three components: Ethereum, zk-GSigproof and smart contract.

1) **Ethereum:** The Ethereum geth[1] comes from Github. We add two interfaces for group signature and zk-SNARK proof verification, which would be called by the smart contract. In order to facilitate the interaction with smart contract and Ethereum, we use web3.py[2] to deploy and call the smart contract.
2) **zk-GSigproof:** We resort to the zk-SNARK algorithm in Github and group signature algorithm [32] to implement zk-GSigproof. In order to generate a zero-knowledge proof that meets our need, we use xJsnark[3] to generate the circuit. xJsnark is a new type of high-level framework, which combines with Jetbrains MPS to provide programmers with a platform for programming in Java. After generating the circuit, we use Jsnark[4] (an open source Java library) to realize key generation, proof generation and verification. We modify the source code of libsnark so that it can take the circuit as an input parameter and support separate proof verification. Finally, we use the CGO to integrate the proof verification function into Ethereum. For group signature algorithm, we implement the scheme for our application scenarios and integrate the group signature verification process into Ethereum.
3) **Smart contract:** First, in order to reflect the decentralization and automated execution of the scheme, we use smart contract to achieve zero-knowledge proof and group signature verification. Then, due to the natural financial properties of smart contracts, we add pre-deposit and settlement functions to it. Finally, we use Solidity[5] as a compilation tool for these functions and modify its source code to support the verification functions of group signature and zk-SNARK proof.

6.2 Performance Evaluation

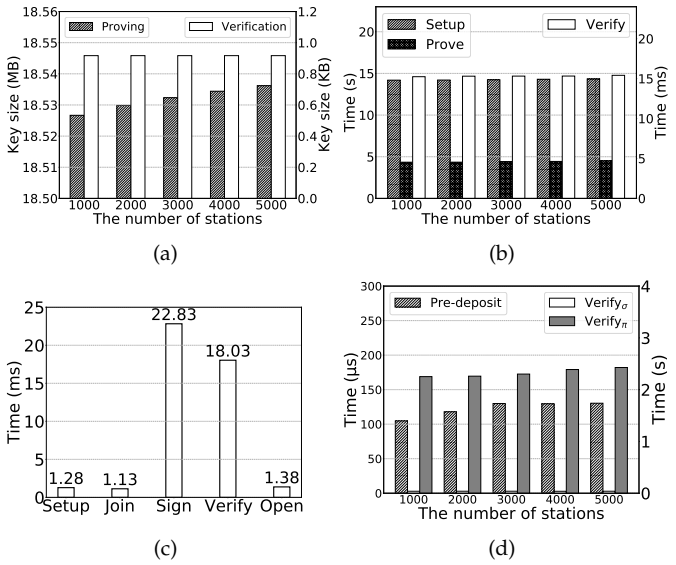

Fig. 6. The performance of zk-SNARK (a) (b), group signature (c) and smart contract (d).

In our experiment, we test the performance of Ethereum, zk-GSigproof and smart contract on local desktops and Aliyun. In order to test the performance of zk-GSigproof more accurately, we conduct experiments on zk-SANRK and the group signature algorithm separately. Each desktop is equipped with Intel Core i5-8500@3.00 GHz*6 and 19.40 GB RAM running 64-bit Ubuntu 18.04. In the experiment on Aliyun, we use 25 ecs.g6.2xlarge instances, each of which running Ubuntu 18.04 system Intel Xeon (Cascade Lake) Platinum 8269CY processor, 8 vCPUs of frequency 2.5/3.2 GHz and 32 GB RAM. We start 4 docker nodes in each

1. https://github.com/ethereum/go-ethereum
2. https://pypi.org/project/web3
3. https://github.com/akosba/xjsnark
4. https://github.com/akosba/jsnark
5. https://github.com/ethereum/solidity

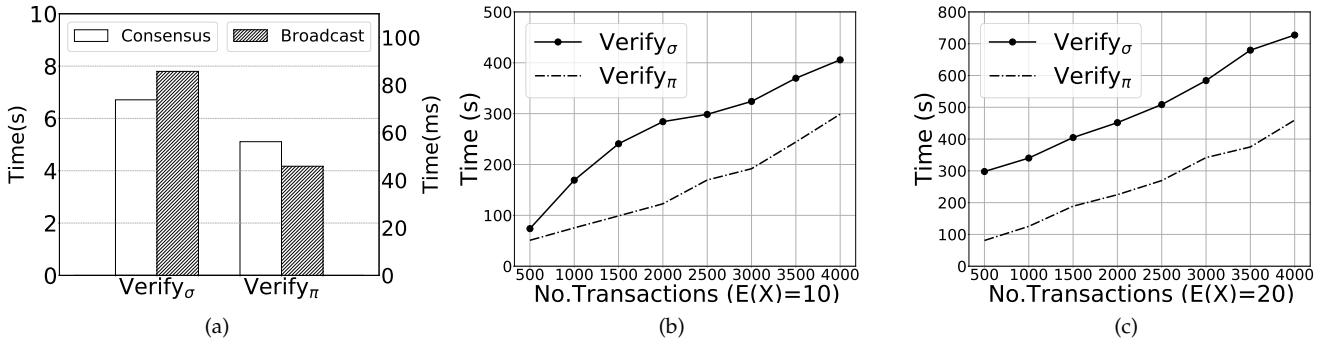

Fig. 7. The transaction delay (a) and Poisson distribution experiment (b) (c) based on a 100-node blockchain in Aliyun.

instance to form a 100-node blockchain network based on the proof-of-work consensus algorithm, and the number of transactions reaches up to 4000.

The performance of zk-SNARK. We test the size of the common reference string crs, i.e. the proving key and the verification key, under different number of stations, and report the results in Fig. 6 (a). One can see that when the number of stations rises from 1000 to 5000, the size increase of the proving and verification keys is small. Specifically, the size of the proving key is about 18.50 MB and that of a verification key is around 0.90 KB. We then test the time of the three algorithms of zk-SNARK, and report the results in Fig. 6 (b). One can observe that the time of setup Π.Setup is about 14.20 seconds, of generating proof Π.Prove is stable at 4.30 seconds, and of verifying proof Π.Verify is the shortest, which is around 15 milliseconds. The above results are reasonable because we fix stations in the circuit instead of taking them as inputs, so the size and running time of zk-SNARK are minimally affected by the number of stations. One may notice that the performance of zk-SNARK (shown in Fig. 6 (a) (b)) is only slightly affected by the number of stations, thus Vehicloak actually has a good scalability.

The performance of group signature. As shown in Fig. 6 (c), the time for each step is at the millisecond level. Specially, the three algorithms, namely member setup (Setup), member join (Join) and open (Open), take 1.2 milliseconds. The generation and verification of group signatures take less than 25 milliseconds. These results all benefit from the optimization of group signature algorithm by schemes [24], [32]. In the verification experiment of group signature and zk-SNARK (shown in Fig. 6 (b) (c)), the experimental results seem to be close to ideal. They are all in milliseconds, which only consume a small amount of computing resources on the chain, proving good performance of zk-GSigproof.

The performance of smart contract. In Fig. 6 (d), we show the execution time of each function in Contract-Vehicloak (Pre-deposit and Validate (including Settlement). In order to make the experimental results more accurate, we further divide Validate into Verify$_\sigma$ (group signature verification of entry/exit) and Verify$_\pi$ (proof verification). One can see that Verify$_\sigma$ only takes about 0.04 seconds and Verify$_\pi$ takes less than 2.50 seconds. It is worth mentioning that the longest time is spent on the transaction deployment, and the time used to verify a proof is very short according to Fig. 6 (b). As the logic of Pre-deposit is relatively simple, the consumed time is at microsecond. To accurately estimate the cost of smart contract, we conduct experiments based on linear regression to simulate the gas of four functions (shown in TABLE 1). Compared to the average cost (21000 gas) in Ethereum, our functions need more gas, which is reasonable, because they include complex calculations such as bilinear pairing computations. Among these functions, Verify$_\sigma$ and Verify$_\pi$ consume more gas, which costs about $2.25. However, there are many feasible methods to reduce the transaction gas, such as zk-Rollup and Optimistic Rollup [33], which can decrease the workload on the main chain by reducing the amount of transaction data or putting the complex work off-chain.

TABLE 1
Smart contract cost of Vehicloak

Function	Gas Used	Ether Cost	USD
Contract Creation	825364	0.000825	1.834
Pre-deposit	62453	0.0000625	0.139
Verify$_\sigma$	709538	0.000710	1.578
Verify$_\pi$	303600	0.000304	0.676

1 Gas = 1 Gwei, and 1 ether=2223 USD.

Ethereum transaction delay. In Fig. 7 (a), we test the broadcast delay and the consensus delay of transactions (calling functions Verify$_\sigma$ and Verify$_\pi$). Broadcast delay refers to the time period of broadcasting a transaction to other peers in the blockchain, while consensus delay refers to the time from when a new transaction is verified until the block containing this transaction is confirmed by miners. In this experiment, all nodes send a total of 4000 transactions. The broadcast delay is less than 90 milliseconds, and the consensus delay is about 4.8 – 6.5 seconds.

Transaction processing with Poisson distribution. In Fig. 7 (b) (c), we show the effect of traffic volume on the experimental results. During this experiment, each transaction is sent 500 – 4000 times, and the transaction generation conforms to Poisson distributions. We set the expectations ($E(X)$) of the Poisson distribution to be 10 and 20. One can see that $E(X)$ = 20 takes more time than $E(X)$ = 10. This indicates that as the frequency of transactions increases, the number of delayed transactions increases. One can also see that when a large number of transactions are sent in a

certain period of time, the delay increases. To remedy this, we present a timeout mechanism in the smart contract to ensure that Vehicloak can avoid transaction aggregation, as detailed in Sec. 5.2.6.

7 Conclusion and Future Research

In this paper, we propose Vehicloak, a decentralized privacy-preserving payment scheme for location-based vehicular services. Vehicloak takes a new cryptographic technique called zk-GSigproof, which can hide the vehicle location, thereby ensuring the authenticity of private information and the correctness of payment. The entire verification process is implemented in smart contract Contract-Vehicloak, which realizes the automation of verification without relying on any trusted third party. In order to prove the practicality of our scheme, we simulate a 100-node blockchain in Aliyun. Experimental results validate the effectiveness of Vehicloak.

In our future research, we will explore the blockchain scalability techniques such as zk-Rollup and Optimistic Rollup, to reduce transaction gas and on-chain workload, which is of great significance for enhancing the performance and feasibility of our scheme. In addition, we will also investigate novel schemes that can handle scenarios in which periodic payments are needed and users may not faithfully report private information.

Acknowledgment

This study was partially supported by the National Key R&D Program of China (No. 2019YFB2102600), the National Natural Science Foundation of China (No. 61832012), the Blockchain Core Technology Strategic Research Program of Ministry of Education of China (2020KJ010301), the Key Research Project of Zhejiang Lab (No. 2022PD0AC01), the Major Basic Research Program of the Shandong Provincial Natural Science Foundation (ZR2020ZD01), and the National Natural Science Foundation of China (61972229).

References

[1] C. Sommer and F. Dressler, *Vehicular Networking*. Cambridge University Press, 2014.

[2] Z. Wan, T. Zhang, W. Liu, M. Wang, and L. Zhu, "Decentralized privacy-preserving fair exchange scheme for v2g based on blockchain," *IEEE Transactions on Dependable and Secure Computing*, 2021.

[3] C. Hu, X. Cheng, Z. Tian, J. Yu, and W. Lv, "Achieving privacy preservation and billing via delayed information release," *IEEE/ACM Transactions on Networking*, 2021.

[4] Z. Guan, Z. Wan, Y. Yang, Y. Zhou, and B. Huang, "Blockmaze: An efficient privacy-preserving account-model blockchain based on zk-snarks," *IEEE Transactions on Dependable and Secure Computing*, 2020.

[5] C. Liu, M. Xu, H. Guo, X. Cheng, Y. Xiao, D. Yu, B. Gong, A. Yerukhimovich, S. Wang, and W. Lv, "Tokoin: A coin-based accountable access control scheme for internet of things," *arXiv preprint arXiv:2011.04919*, 2020.

[6] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Business Review*, p. 21260, 2008.

[7] C. Liu, H. Guo, M. Xu, S. Wang, D. Yu, J. Yu, and X. Cheng, "Extending on-chain trust to off-chain – trustworthy blockchain data collection using trusted execution environment (tee)," *IEEE Transactions on Computers*.

[8] M. Xu, F. Zhao, Y. Zou, C. Liu, X. Cheng, and F. Dressler, "Blown: A blockchain protocol for single-hop wireless networks under adversarial sinr," *IEEE Transactions on Mobile computing*, 2022.

[9] M. Xu, C. Liu, Y. Zou, F. Zhao, J. Yu, and X. Cheng, "wchain: A fast fault-tolerant blockchain protocol for multihop wireless networks," *IEEE Transactions on Wireless Communications*, vol. 20, pp. 6915–6926, October 2021.

[10] M. Xu, S. Liu, D. Yu, X. Cheng, S. Guo, and J. Yu, "Cloudchain: A cloud blockchain using shared memeory consensus and rdma," *IEEE Transactions on Computers*.

[11] M. Belotti, N. Božić, G. Pujolle, and S. Secci, "A vademecum on blockchain technologies: When, which, and how," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3796–3838, 2019.

[12] R. A. Popa, H. Balakrishnan, and A. J. Blumberg, "Vpriv: Protecting privacy in location-based vehicular services," in *Usenix Security Symposium*, 2009.

[13] J. Balasch, A. Rial, C. Troncoso, B. Preneel, I. Verbauwhede, and C. Geuens, "Pretp: Privacy-preserving electronic toll pricing.," in *USENIX Security Symposium*, 2010.

[14] S. Meiklejohn, K. Mowery, S. Checkoway, and H. Shacham, "The phantom tollbooth: Privacy-preserving electronic toll collection in the presence of driver collusion.," in *USENIX security symposium*, vol. 201, 2011.

[15] V. Fetzer, M. Hoffmann, M. Nagel, A. Rupp, and R. Schwerdt, "P4tc—provably-secure yet practical privacy-preserving toll collection," *Proceedings on Privacy Enhancing Technologies*, no. 3, pp. 62–152, 2020.

[16] T. Salman, M. Zolanvari, A. Erbad, R. Jain, and M. Samaka, "Security services using blockchains: A state of the art survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 858–880, 2018.

[17] A. R. Shahid, N. Pissinou, L. Njilla, S. Alemany, A. Imteaj, K. Makki, and E. Aguilar, "Quantifying location privacy in permissioned blockchain-based internet of things (iot)," in *Proceedings of the 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, pp. 116–125, 2019.

[18] H. Li, L. Pei, D. Liao, G. Sun, and D. Xu, "Blockchain meets vanet: An architecture for identity and location privacy protection in vanet," *Peer-to-Peer Networking and Applications*, vol. 12, no. 5, pp. 1178–1193, 2019.

[19] H. Shen, J. Zhou, Z. Cao, X. Dong, and K.-K. R. Choo, "Blockchain-based lightweight certificate authority for efficient privacy-preserving location-based service in vehicular social networks," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6610–6622, 2020.

[20] D. Chaum and E. Van Heyst, "Group signatures," in *Workshop on the Theory and Application of of Cryptographic Techniques*, pp. 257–265, Springer, 1991.

[21] O. Goldreich and Y. Oren, "Definitions and properties of zero-knowledge proof systems," *Journal of Cryptology*, vol. 7, no. 1, pp. 1–32, 1994.

[22] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *2014 IEEE Symposium on Security and Privacy*, pp. 459–474, IEEE, 2014.

[23] B. Parno, J. Howell, C. Gentry, and M. Raykova, "Pinocchio: Nearly practical verifiable computation," in *2013 IEEE Symposium on Security and Privacy*, pp. 238–252, IEEE, 2013.

[24] J. Camenisch and J. Groth, "Group signatures: Better efficiency and new theoretical aspects," in *SCN 2004*, vol. 3352 of *Lecture Notes in Computer Science*, pp. 120–133, 2004.

[25] L. Blum, M. Blum, and M. Shub, "A simple unpredictable pseudo-random number generator," *SIAM Journal on Computing*, vol. 15, no. 2, pp. 364–383, 1986.

[26] A. Liu and P. Ning, "Tinyecc: A configurable library for elliptic curve cryptography in wireless sensor networks," in *2008 International Conference on Information Processing in Sensor Networks (ipsn 2008)*, pp. 245–256, IEEE, 2008.

[27] J. Groth, "On the size of pairing-based non-interactive arguments," in *Annual international conference on the theory and applications of cryptographic techniques*, pp. 305–326, Springer, 2016.

[28] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev, "Scalable, transparent, and post-quantum secure computational integrity.," *IACR Cryptol. ePrint Arch.*, vol. 2018, p. 46, 2018.

[29] A. Gabizon, Z. J. Williamson, and O. Ciobotaru, "Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge.," *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 953, 2019.

[30] W. Shao, C. Jia, Y. Xu, K. Qiu, Y. Gao, and Y. He, "Attrichain: Decentralized traceable anonymous identities in privacy-preserving

permissioned blockchain," *Computers & Security*, vol. 99, p. 102069, 2020.
[31] B. Bünz, S. Agrawal, M. Zamani, and D. Boneh, "Zether: Towards privacy in a smart contract world," in *International Conference on Financial Cryptography and Data Security*, pp. 423–443, Springer, 2020.
[32] N. P. Smart and B. Warinschi, "Identity based group signatures from hierarchical identity-based encryption," in *International Conference on Pairing-Based Cryptography*, pp. 150–170, Springer, 2009.
[33] O. Marukhnenko and G. Khalimov, "The overview of decentralized systems scaling methods," *COMPUTER AND INFORMATION SYSTEMS AND TECHNOLOGIES*, 2021.

Yihao Guo is currently pursuing a Ph.D. degree in the School of Computer Science and Technology, Shandong University, China. He obtained his bachelor degree in Computer Science and Technology from Qufu Normal University in 2019. His research interests include blockchain, distributed computing, applied cryptography and security.

Zhiguo Wan is a principal investigator in the Zhejiang Lab, Hangzhou, Zhejiang, China. His main research interests include security and privacy for cloud computing, Internet-of-Things and blockchain. He received his B.S. degree in computer science from Tsinghua University in 2002, and Ph.D. degree in information security from National University of Singapore in 2007. He was a postdoc in Katholieke University of Leuven, Belgium and an assistant professor in the School of Software, Tsinghua University, Beijing, China.

Hui Cui received her Ph.D. degree in the School of Computing and Information Technology, University of Wollongong, Australia. She then worked as a research fellow in the Secure Mobile Centre under the School of Information Systems, Singapore Management University, Singapore. Then, she worked as a research fellow in the School of Science, RMIT University, Melbourne, Australia and Data61, CSIRO, Melbourne, Australia. Now, she is a lecturer in the Murdoch University, Perth, Australia.

Xiuzhen Cheng received her MS and PhD degrees in computer science from University of Minnesota, Twin Cities, in 2000 and 2002, respectively. She was a faculty member at the Department of Computer Science, George Washington University, from 2002-2020. Currently she is a professor of computer science at Shandong University, Qingdao, China. Her research focuses on blockchain computing, security and privacy, and Internet of Things. She is a Fellow of IEEE and a Fellow of CSEE.

Falko Dressler (Fellow, IEEE) is full professor and Chair for Telecommunication Networks at the School of Electrical Engineering and Computer Science, TU Berlin. He received his M.Sc. and Ph.D. degrees from the Dept. of Computer Science, University of Erlangen in 1998 and 2003, respectively. Dr. Dressler has been associate editor-in-chief for IEEE Trans. on Mobile Computing and Elsevier Computer Communications as well as an editor for journals such as IEEE/ACM Trans. on Networking, IEEE Trans. on Network Science and Engineering, Elsevier Ad Hoc Networks, and Elsevier Nano Communication Networks. He has been chairing conferences such as IEEE INFOCOM, ACM MobiSys, ACM MobiHoc, IEEE VNC, IEEE GLOBECOM. He authored the textbooks Self-Organization in Sensor and Actor Networks published by Wiley & Sons and Vehicular Networking published by Cambridge University Press. He has been an IEEE Distinguished Lecturer as well as an ACM Distinguished Speaker. Dr. Dressler is an IEEE Fellow as well as an ACM Distinguished Member. He is a member of the German National Academy of Science and Engineering (acatech). He has been serving on the IEEE COMSOC Conference Council and the ACM SIGMOBILE Executive Committee. His research objectives include adaptive wireless networking (sub-6GHz, mmWave, visible light, molecular communication) and wireless-based sensing with applications in ad hoc and sensor networks, the Internet of Things, and Cyber-Physical Systems.