Blockchain Based Zero-Knowledge Proof of Location in IoT

Wei Wu, Erwu Liu, Xinglin Gong and Rui Wang
College of Electronics and Information Engineering, Tongji University, Shanghai, China
Emails: {1832912, erwuliu, xinglingong, ruiwang}@tongji.edu.cn

***Abstract*—With the development of precise positioning technology, a growing number of location-based services (LBSs) facilitate people's life. Most LBSs require proof of location (PoL) to prove that the user satisfies the service requirement, which exposes the user's privacy. In this paper, we propose a zero-knowledge proof of location (zk-PoL) protocol to better protect the user's privacy. With the zk-PoL protocol, the user can choose necessary information to expose to the server, so that hierarchical privacy protection can be achieved. The evaluation shows that the zk-PoL has excellent security to resist main attacks, moreover the computational efficiency is independent of input parameters and the zk-PoL is appropriate to delay-tolerant LBSs.**

***Index Terms*—Blockchain, zero-knowledge proof, location-based service, zk-PoL, IoT**

I. Introduction

In recent years, the popularity of smart terminal devices and the development of sophisticated high-precision position sensors have led to many location based-services (LBSs) [1], such as location-based rewards and digging services, location-based recommendations and location-based social network (LBSN) [2]. Most LBSs require the user's proof of location (PoL) to prove that the user satisfies service requirements. For example, a user needs to prove that he is in the mart to obtain a discount coupon. While enjoying the convenience of these services, the user's location privacy is always revealed through the PoL to the server or malicious adversaries. Hence, when the user provides PoL to the server, the user's privacy must be protected.

A large number of related works on PoL emerge to protect the user's privacy. The previously proposed PoL system architectures can be roughly divided into two types: centralized and distributed. In the former, the user data is stored in the central server, while the latter one applies distributed storage. Tyagi et al. [3] propose a location privacy architecture based on the approach like a mix zone model with other approaches to protect location privacy. Javali et al. [4], based on channel state information (CSI) and fuzzy vault, a cryptographic primitive, propose a location proof generation and verification scheme in which provers can prove their presence in particular time with high confidence. Li et al. [5], relying on the existing WiFi or cellular network access points (APs), propose an infrastructure-based scheme to provide privacy-preserving location proof, in which the database can verify the location without knowing the user's accurate location. All the above solutions perform in a centralized way, which would lead to the following shortcomings. First, a powerful dedicated central server is needed to serve a large number of nodes, which results in an unbalanced load between the server and the node. Once the server is overloaded, the entire system will be paralyzed. Second, a malicious adversary is easier to capture the system because it has a single opponent, the central server. Once the central server is captured, the entire system is captured. Third, the user's data is stored in the central server and only the server can control them. This will lead to the user's losing control of his data.

In order to solve the problem of centralized architecture, Amoretti et al. [6] propose a blockchain-based PoL system, in which users access a blockchain network in a self-organized mode so that the users can communicate with neighboring $witnesses$ and request them to issue location certificates.

Although this solution does not require a dedicated IoT device, the security will be compromised because it does not restrict mobile nodes' access to the network and the permission of the nodes already accessed in the network is also unrestricted. Therefore, combined with blockchain technology and zero-knowledge proof, we propose a zero-knowledge proof of location (zk-PoL) protocol. Our zk-PoL protocol consists of two parts: the generation scheme and verification scheme. By generation scheme, an approved user could obtain a location certificate from the smart AP. Here 'smart' means that smart contact can be embedded in the AP to regulate AP's behavior and process data. By verification scheme, the user could use the location certificate to generate a zero-knowledge proof of location for LBSs he needs. Moreover, this zk-PoL protocol allows the user to achieve hierarchical privacy protection by generating different proofs with different input parameters. We also analyze the performance of the zk-PoL protocol in security and computational efficiency. The security analysis shows that the zk-PoL is able to resist the main attacks. The experiment demonstrates that the system performance is independent of input parameters and the zk-PoL is applicable to delay-tolerant LBSs.

The remainder is organized as follows. In Section II, we introduce the essentials of blockchain and zero-knowledge proof. In Section III, we build our system model and based

This work is supported in part by the grants from the National Science Foundation of China (No. 61571330), Shanghai Integrated Military and Cilivian Development Fund (No. JMRH-2018-1075), and Science and Technology Commission of Shanghai Municipality (No. 19511102002). Corresponding author: Erwu Liu.

978-1-7281-5089-5/20/$31.00 ©2020 IEEE

on it, we give our zk-PoL protocol. In Section IV we analyze the security and computational efficiency of our system. In Section V, we conclude the work of this paper.

II. Preliminary

This section briefly introduces the knowledge of blockchain technology and zero-knowledge proof.

A. Blockchain Technology

Blockchain technology is a fusion of multiple technologies, including encryption, consensus mechanisms, incentives, distributed storage and smart contracts [7]–[9]. In essence, a blockchain is a chained public ledger guaranteed to be indestructible with cryptographic methods. It consists of a series of blocks, and the latter block is connected to the previous one by including the cryptographic hash of the previous block. In addition, the block also contains timestamps, nonce, transaction data, etc. Its structure is shown in Fig. 1. Then, by the consensus algorithm, such as Proof of

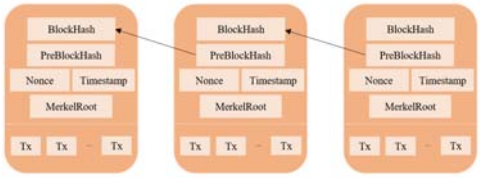

Fig. 1. Structure of A Simplified Block Chain

Work (PoW) and Proof of Stake (PoS), all peers in the network agree on the ownership of the bookkeeping right. The peer that obtains the bookkeeping right adds the packaged block to the chain afterward. The data recorded in the chain cannot be modified. This whole process is defined as mining, and the peers participating in the bookkeeping right competition are defined as miners. For the public chain, the miner that finally obtains the bookkeeping right and completes the bookkeeping will receive a token reward. This mechanism is referred to as the incentive mechanism. Currently, there are at least four types of blockchain networks: public blockchain, private blockchain, consortium blockchain, and hybrid blockchain. The blockchain network introduced in this paper is the hybrid chain, in which different peers have different permission according to their diverse responsibilities.

B. Zero-Knowledge Proof

The zero-knowledge proof is a kind of method by which a $prover$ can convince a $verifier$ that some statement is true without revealing any other information. Any zero-knowledge proof protocol should satisfy three properties [10]:

- Completeness: If the statement is true then a $prover$ can convince a $verifier$.
- Soundness: A cheating $prover$ can not convince a $verifier$ of a false statement.
- Zero-knowledge: The interaction only reveals whether a statement is true or not and nothing else.

Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (zk-SNARK) is a novel form of zero-knowledge cryptography which is introduced in [11] and built in [12]. Quadratic Arithmetic Program (QAP) [13] and the Pinocchio protocol [14] make it more effective and universal. Its first widespread application is Zcash [15]. The following is a brief introduction to the zk-SNARK protocol [10].

Any complex computation or program can be regarded as one consisting of a series of simple operations which are defined as

$$left\ operand\ \textbf{operator}\ right\ operand = output, \tag{1}$$

where $operator \in \{+, -, \cdot, \div\}$. Naturally, if we would like to verify the correctness of the entire computation result, we just need to verify the correctness of the results of each operation separately, but such verification is very inefficient. Therefore, we can interpolate these operations into a polynomial to verify all operations at a time, that is, to verify polynomial

$$L(x)\ \textbf{operator}\ R(x) = O(x), \tag{2}$$

where $L(x)$, $R(x)$ and $O(x)$ represent left, right and output polynomials, respectively. This is the basic idea of QAP. Here, the **operator** is rewritten as a multiplication since the addition and subtraction are made in each operand, and the division can be converted into multiplication. It assumes that the computation can be decomposed into d operations. If the computation is correct, the equation is true for each $x = i,\ i \in \{1, ..., d\}$. Therefore, polynomial $L(x) \cdot R(x) - O(x) = 0$ should have roots $x = i,\ i \in \{1, ..., d\}$, which is denoted as

$$p(x) = L(x) \cdot R(x) - O(x) = t(x) \cdot h(x), \tag{3}$$

where $t(x) = \prod_{i=1}^{d} (x - i)$.

In this way, by checking whether the polynomial $p(x)$ has the factor $t(x)$, the correctness of the computation can be checked. If a $prover$ intends to prove the correctness of the computation, he only needs to provide $L(x)$, $R(x)$, $O(x)$ and $h(x)$ which satisfy (3). In order to remain the same variable assignments of the same operands (or output) in different operations unchanged, $L(x)$, $R(x)$ and $O(x)$ are rewritten as the sum of the variable polynomials

$$\begin{cases} L(x) = \sum_{i=0}^{n} v_i l_i(x) \\ R(x) = \sum_{i=0}^{n} v_i r_i(x) \\ O(x) = \sum_{i=0}^{n} v_i o_i(x) \end{cases}, \tag{4}$$

where $i \in \{0, ..., n\}$. $l_i(x)$ is the ith variable polynomial of the left operand; n is the number of variables involved; $l_0(x)$ is variable polynomial of pseudo-variable one. The variable polynomials are determined by the specific computation. The $prover$ cannot modify these variable polynomials but only assign them, that is, multiply them with v_i. Knowledge-of-Exponent Assumption (KEA) is utilized to construct variable consistency polynomial. Variable consistency polynomial ensures variable assignments consistency across operands in the same operation, and it is denoted as

$$\begin{aligned} Z(x) =& v_i \sum_{i=1}^{n} \beta_l l_i(x) + \beta_r r_i(x) + \beta_o o_i(x) \\ =& \beta_l L(x) + \beta_r R(x) + \beta_o O(x), \end{aligned} \tag{5}$$

where β_l, β_r and β_o are random numbers. Next, the $prover$ has to convince the $verifier$ that he knows the variable assignments that satisfy the restrictions, but the $prover$ cannot directly inform the $verifier$ of the variable assignments that satisfy (3). One reason is that it will expose all information to $verifier$, the other is that the multiplication of polynomials is extremely complex. According to the Fundamental Theorem of Algebra, for any polynomial, we can use its evaluation at any point to represent its unique identity. Therefore, the $verifier$ can verify whether the evaluation of (3) is satisfied at a random point s, that is, verify

$$L(s) \cdot R(s) - O(s) = t(s) \cdot h(s). \tag{6}$$

In order to ensure that the $prover$ cannot forge the polynomial, the random point s should be encrypted. So, the $verifier$ needs to verify (6) in the encryption, that is, to verify

$$e(g^{L(s)}, g^{R(s)}) = e(g^{t(s)}, g^{h(s)}) \cdot e(g^{O(s)}, g). \tag{7}$$

In addition, the $verifier$ also has to validate the constructing of polynomials by checking

$$\begin{cases} e(g^{L(s)}, g^{\alpha_l}) = e(g^{\alpha_l L(s)}, g) \\ e(g^{R(s)}, g^{\alpha_r}) = e(g^{\alpha_r R(s)}, g) \\ e(g^{O(s)}, g^{\alpha_o}) = e(g^{\alpha_o O(s)}, g) \end{cases}, \tag{8}$$

$$e(g^{L(s)}, g^{\beta_l}) \cdot e(g^{R(s)}, g^{\beta_r}) \cdot e(g^{O(s)}, g^{\beta_o}) = e(g^{Z(s)}, g), \tag{9}$$

where g is the generation point of elliptic curves and $e(g^*, g^*)$ is cryptography pairings (or bilinear map), which maps two encrypted inputs from two encrypted space to their multiplied form in different encrypted space, as shown in Fig. 2.

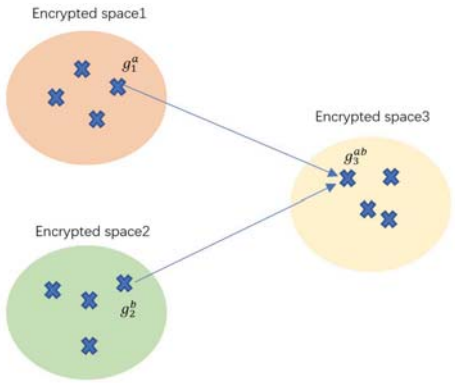

Fig. 2. Schematic Diagram of Bilinear Map

If (7), (8) and (9) are satisfied, the $verifier$ can believe that the $prover$ indeed knows true variable assignments with great confidence and do not know any other information about these variable assignments.

III. System Model

The location data depict someone's digital trace which gives more contextual information such as individuals' habits, interests, activities and relationships. This information exposes the user to an unwanted advertisement and location-based spams/scams, causing social reputation or economic damage and making them victims of blackmail or even physical violence [16]. Moreover, location data are insecurely transmitted and stored, and uncontrollable by its owner in traditional mode. In order to solve these drawbacks, we consider a system model with smart IoT devices. Our system model contains three components: the server, the user, and the ledger, as shown in Fig. 3, where the ledger is a blockchain network consisting of smart access points (APs). The general function of each component is described below.

Fig. 3. System Model of zk-PoL

APs are responsible for issuing location certificate to the user, providing the user with Common Reference String (CRS) for generating the proof and saving the service record, i.e. which service has been responded.

The user requests the nearby AP to issue a location certificate with short-range communication mode. When some LBS is needed, the user uses the location certificate to generate a proof for this LBS.

The server receives the service request and the proof sent by the user. After validating the proof, the server provides the user with the corresponding LBS. Once the service is obtained by the user, the server will request the AP to save the service record, updating the ledger.

This kind of system enables separation of the user's data and the service. The public ledger only holds the root of trust of data, and the actual data is stored locally by the user. When the user requests for LBSs from the server, he just needs to prove the truth and validity of location data that he holds.

Focusing on the interaction details among all system components, we propose the zk-PoL protocol. Our protocol consists of two parts: the generation and verification scheme, where generation scheme mainly describes the issuance of the location certificate, and verification scheme mainly describes the generation and verification of the proof. The general process of the protocol is shown in Fig. 4 and the details are given as follows.

A. Generation scheme

Assuming that $user_i$ needs to request for LBSs from the $server_j$. The first thing he should do is to access to the

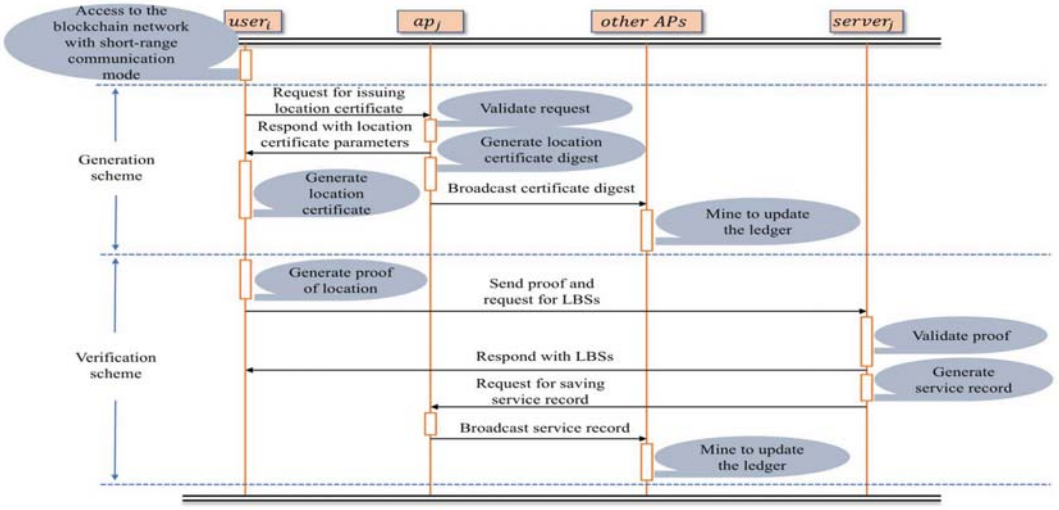

Fig. 4. Process Diagram of zk-PoL

blockchain network and communicate with the nearby access point ap_j with short-range communication mode, such as WiFi, bluetooth and ZigBee. Then the $user_i$ should request for location certificate issuing from the ap_j, and the request is denoted as

$$req_{i\to j}: \left\{ inf_i: \left\{ \begin{array}{c} pk^i_{user} \\ \langle longitude, latitude\rangle^i_{user} \end{array} \right\}, \left\{ Hash\left(inf_i\right)\right\}_{sk^i_{user}} \right\}_{pk^j_{ap}}, \tag{10}$$

where $req_{i\to j}$ denotes the request sent by peer i to peer j. The sk^i_{user} denotes the private key of the $user_i$; the pk^i_{user} and pk^j_{ap} denote the public key (also the identity in the blockchain network) of $user_i$ and ap_j, respectively. It is worth noting that the user equipment has no correspondence with ID in the blockchain network. The blockchain network ID only matches the private key. The $\langle longitude, latitude\rangle^i_{user}$ denotes the geographic coordinates of $user_i$. The $user_i$ uses sk^i_{user} to sign the inf_i to prevent the malicious adversary from tampering with it, and uses pk^j_{ap} to encrypt the $req_{i\to j}$ to ensure that only the ap_j can correctly decrypt it.

After receiving $req_{i\to j}$, the ap_j processes it according to the following rules:

- If the $req_{i\to j}$ has been tampered with, it will bediscarded.
- If the geographic coordinates in inf_i is out of the range of short-range communication, the $req_{i\to j}$ is discarded.
- Otherwise, the ap_j responds to the $user_i$.

The response is defined as

$$res_{j\to i}: \left\{ inf_j: \left\{ \begin{array}{c} rand^i_{user} \\ time^i_{user} \end{array} \right\}, \left\{ Hash\left(inf_j\right)\right\}_{sk^j_{ap}} \right\}_{pk^i_{user}}, \tag{11}$$

where $rand^i_{user}$ is a random number and can be regarded as a serial number of certificate. The $time^i_{user}$ indicates the moment when the certificate is generated. Similarly, the ap_j uses its own private key to sign the inf_j, and uses $user_i$'s public key to encrypt the $res_{j\to i}$. Meanwhile, with inf_i and inf_j, the ap_j generates a digest of location certificate dig_i

$$dig_i: \left\{ Hash\left(\begin{array}{c} pk^i_{user} \\ \langle longitude, latitude\rangle^i_{user} \\ rand^i_{user}, time^i_{user} \end{array} \right) \right\}. \tag{12}$$

Then, the dig_i is broadcast to miners (APs) in the blockchain network.

The $user_i$ checks whether the $res_{j\to i}$ has been tampered with or not. If it is, the $user_i$ discards the $res_{j\to i}$. If not, with inf_i and inf_j, the $user_i$ generates cer_i

$$cer_i: \left\{ \begin{array}{c} pk^i_{user} \\ \langle longitude, latitude\rangle^i_{user} \\ rand^i_{user}, time^i_{user} \end{array} \right\}. \tag{13}$$

After cer_i is generated, the ap_j will delete inf_i and inf_j according to the embedded smart contract. Meanwhile, the miners package the received dig_i into their own blocks. Then by mining, one AP is selected to add its block to the blockchain, updating the ledger. Once dig_i is recorded in the ledger, it cannot be tampered with. The $user_i$ can generate a proof proving that his cer_i is in accordance with the dig_i in the ledger to obtain related LBSs from the $server_j$.

B. Verification scheme

Due to the different service requirements of different LBSs, when the $user_i$ requests for LBSs from the $server_j$, he needs to choose different input parameters, as shown in TABLE I, to generate the appropriate proof proving that he satisfies the service requirement.

The four levels of input parameters expose different necessary information to the server, achieving so-called hierarchical privacy protection, which can be applied to different scenarios:

- Level 1: The $user_i$ locally generates a proof with input parameters in level 1. This can be used to prove that someone has appeared at certain time in the range

TABLE I
INPUT PARAMETERS

Level	Public Parameters	Private Parameters
1	$null$	pk^i_{user}, $\langle longitude, latitude\rangle^i_{user}$, $rand^i_{user}$, $time^i_{user}$
2	$\langle longitude, latitude\rangle^i_{user}$	pk^i_{user}, $rand^i_{user}$, $time^i_{user}$
3	$\langle longitude, latitude\rangle^i_{user}$, $time^i_{user}$	pk^i_{user}, $rand^i_{user}$
4	pk^i_{user}, $\langle longitude, latitude\rangle^i_{user}$, $time^i_{user}$	$rand^i_{user}$

covered by APs and be applied to the scenarios such as authorization of attractions review where the critic must prove that he has been some tourist area in order to get commenting permission.

- Level 2: The $user_i$ locally generates a proof with input parameters in level 2, disclosing its own coordinate. This can be used to prove that someone has appeared at a particular location and certain moment, and can be applied to the scenarios such as fixed-point Punching for Coupon Service (PCS) where the user needs to punch in the specific place in order to get the coupon.
- Level 3: The $user_i$ locally generates a proof with input parameters in level 3, disclosing its own coordinate and time. This can be used to prove that someone has appeared at a particular location and particular moment, and can be applied to scenarios such as real-time location based recommendations where the user must prove that he is now at a specific location in order to get precise recommendations.
- Level 4: The $user_i$ locally generates a proof with input parameters in level 4, disclosing its own coordinate, time, and identity, i.e. pk^i_{user}. This can be used to prove that a particular person has appeared at a particular place and a particular moment. For example, a suspect needs the alibi to prove that he is not at the crime scene at a particular moment.

Without loss of generality, the following takes the PCS as an example to illustrate the details of the verification scheme. In PCS, the $user_i$ needs to punch in the specific place to get the coupon. Note that the ID of $user_i$ when he requests LBSs from the $server_j$ is no longer pk^i_{user}, so the $user_i$'s identity is not exposed. The $user_i$ should use the location certificate cer_i to generate a relevant proof and then send it to the $server_j$ for PCS. The specific process goes as following.

1) Computation to QAP: As mentioned earlier, any complex computation can be converted into QAP. The complex computations in the PCS case is

$$\begin{cases} Hash(rand^i_{user}) = hr_i \\ Hash\left(pubpara, pripara\right) = dig_i \end{cases}, \tag{14}$$

where hr is the hash of serial number and it is set for special single-use service. Its specific role will be discussed below. The $pubpara$ and $pripara$ are public parameters and private parameters in TABLE I, respectively. The QAP obtained from computation is

$$\left\{\left\{l_i\left(x\right), r_i\left(x\right), o_i\left(x\right)\right\}, t\left(x\right)\right\} \quad i \in \left\{0, ...m, ..., n\right\}, \tag{15}$$

where $l_i\left(x\right)$, $r_i\left(x\right)$ and $o_i\left(x\right)$ are variable polynomials; n is the number of variables; $l_0\left(x\right)$, $r_0\left(x\right)$ and $o_0\left(x\right)$ are polynomials of pseudo-variable one and $l_1\left(x\right)$ to $l_m\left(x\right)$, $r_1\left(x\right)$ to $r_m\left(x\right)$ and $o_1\left(x\right)$ to $o_m\left(x\right)$ are polynomials of public variables. In this case $m = 1$, which represents the public parameter $\langle longitude, latitude\rangle^i_{user}$; $t\left(x\right)$ is the target polynomial and its degree d denotes the number of operations.

2) Setup: After the conversion to QAP is completed, some randoms $\langle s, g, \rho_l, \rho_r, \alpha_l, \alpha_r, \alpha_o, \beta, \gamma\rangle$ are needed. With these randoms, the APs generate proving key

$$proving \quad key: \left\{ \begin{array}{c} \left\{g^{s^i}\right\} \\ \left\{g_l^{l_j(s)}, g_r^{r_j(s)}, g_o^{o_j(s)}\right\} \\ \left\{g_l^{\alpha_l l_j(s)}, g_r^{\alpha_r r_j(s)}, g_o^{\alpha_o o_j(s)}\right\} \\ \left\{g_l^{\beta l_j(s)} g_r^{\beta r_j(s)} g_o^{\beta o_j(s)}\right\} \end{array} \right\}, \tag{16}$$

where $g_l = g^{\rho_l}$, $g_r = g^{\rho_r}$, $g_o = g^{\rho_l \cdot \rho_r}$, $i \in \left\{1, ..., d\right\}$ and $j \in \left\{m+1, ..., n\right\}$, and verification key

$$verification \quad key: \left\{ \begin{array}{c} g^1, g_o^{t(s)} \\ \left\{g_l^{l_i(s)}, g_r^{r_i(s)}, g_o^{o_i(s)}\right\} \\ g^{\alpha_l}, g^{\alpha_r}, g^{\alpha_o}, g^{\gamma}, g^{\gamma\beta} \end{array} \right\}, \tag{17}$$

where $i \in \left\{0, ..., m\right\}$.

Once proving key and verification key are generated, these randoms are deleted by APs. Any adversary with these randoms has the ability to falsify a false proof to cheat the server. This is why these randoms are called 'toxic waste'.

The above two phases can be collectively referred to as Trusted Setup (TS) for generating CRS, i.e., the pair of proving and verification keys. In our protocol, TS is executed by trusted APs at the system initialization.

3) Calculate witness: In order to generate the proof, the $user_i$ needs to calculate all the intermediate variables $witness$

$$witness: \left\{v_{m+1}, ..., v_n\right\}. \tag{18}$$

4) Generate proof: The $user_i$ calculates polynomials $L\left(x\right)$, $R\left(x\right)$ and $O\left(x\right)$ through assigning v_i to $l_i\left(x\right)$, $r_i\left(x\right)$ and $o_i\left(x\right)$, as shown in (4), and then obtains $h\left(x\right)$

$$h\left(x\right) = \frac{L\left(x\right)R\left(x\right) - O\left(x\right)}{t\left(x\right)}. \tag{19}$$

Then using the proving key, the $user_i$ calculates

$$g_l^{L_p(s)} = g_l^{\sum_{i=m+1}^{n} v_i l_i(s)}, \tag{20}$$

$$g_l^{L'_p(s)} = g_l^{\alpha_l \sum_{i=m+1}^{n} v_i l_i(s)}. \tag{21}$$

Similarly, $g_r^{R_p(s)}$, $g_r^{R'_p(s)}$, $g_o^{O_p(s)}$ and $g_o^{O'_p(s)}$ can be obtained, and the variable consistency polynomial is calculated by

$$\begin{aligned} g^{Z(s)} &= \prod_{i=m+1}^{n} g_l^{v_i \beta l_i(s)} g_r^{v_i \beta r_i(s)} g_o^{v_i \beta o_i(s)} \\ &= g_l^{\beta L_p(s)} g_r^{\beta R_p(s)} g_o^{\beta O_p(s)} \end{aligned} \tag{22}$$

In this way, the $user_i$ obtains the proof pro_i

$$pro_i : \left\{ \begin{matrix} g_l^{L_p(s)}, g_l^{L_p^{'}(s)}, g_o^{R_p(s)}, g_r^{R_p^{'}(s)}, \\ g_o^{O_p(s)}, g_o^{O_p^{'}(s)}, g^{Z(s)}, g^{h(s)} \end{matrix} \right\}. \tag{23}$$

After that, the $user_i$ sends proof to the $server_j$ and requests the PCS from the $server_j$, and the request is

$$req_{i \to j} : \{pro_i, ind_{PCS}, hr_i\}, \tag{24}$$

where ind_{PCS} is the index of PCS which $user_i$ requested.

5) Verify proof: The $server_j$ checks if the service record has already existed in the ledger. If it is, the request for PCS is denied. If not, using the verification key, the $server_j$ validate pro_i by checking

$$\begin{cases} e\left(g_l^{L_p(s)}, g^{\alpha_l}\right) = e\left(g_l^{L_p^{'}(s)}, g\right) \\ e\left(g_r^{R_p(s)}, g^{\alpha_r}\right) = e\left(g_r^{R_p^{'}(s)}, g\right) \\ e\left(g_o^{O_p(s)}, g^{\alpha_o}\right) = e\left(g_o^{O_p^{'}(s)}, g\right) \end{cases}, \tag{25}$$

$$e\left(g_l^{L_p(s)} g_r^{R_p(s)} g_o^{O_p(s)}, g^{\beta\gamma}\right) = e\left(g^{Z(s)}, g^{\gamma}\right), \tag{26}$$

$$\begin{aligned} & e\left(g_l^{L_v(s)+L_p(s)}, g_r^{R_v(s)+R_p(s)}\right) \\ & = e\left(g_o^{t(s)}, g^{h(s)}\right) \cdot e\left(g_o^{O_v(s)+O_p(s)}, g\right) \end{aligned}, \tag{27}$$

where $g_l^{L_v(s)} = g_l^{l_0(s)+\ldots+v_m l_m(s)}$, and similarly for $g_r^{R_v(s)}$ and $g_o^{O_v(s)}$. After that, the $server_j$ provides PCS for the $user_i$ and sends a service record rec_{ji} to the ap_j, which is defined as

$$rec_{ji} : \left\{pk_{sever}^{j}, ind_{PCS}, hr_i\right\}. \tag{28}$$

Then rec_{ji} is broadcasted to other APs in the blockchain network and saved in the ledger through mining, which indicates that the $server_j$ has provided PCS for the $user_i$. When the $user_i$ attempts to request PCS again to obtain another coupon, the $server_j$ can reject the request because the service record has already existed in the ledger.

At this point, we have introduced the complete zk-PoL with input parameters in level 1. The zk-PoL with input parameters in other levels has some subtle differences in generating CRS, calculating witness, generating proof and verifying proof, but the general process is the same. Because of space limitation, the details will not be discussed here.

IV. Evaluation

In this section, we analyze the proposed system security to resist main attacks, including cheating on certificate parameters, cheating on service and privacy inference attack. Moreover, we will also discuss computational efficiency according to the experiment.

A. Security

1) Cheating on Certificate Parameters: In generation scheme, the certificate parameters $time$ and $rand$ are generated by the trusted APs, so these two parameters can not be falsified. A malicious user can only cheat on $\langle longitude, latitude \rangle_{user}$, and pk_{user}. However, by detecting whether the coordinate sent by the malicious peer is in the range of short-range communication, the location cheating can be prevented. In addition, to prevent cheating on pk_{user}, the AP checks

$$D_{pk_{user}^{'}}\left(E_{sk_{user}}\left(Hash\left(inf\right)\right)\right) = Hash\left(inf\right), \tag{29}$$

where D and E are encryption algorithms and decryption algorithms of asymmetric cryptography, respectively. The $pk_{user}^{'}$ is the ID to be checked.

2) Cheating on Services: In the verification scheme, the server, by checking variable polynomials restriction, variable values consistency and valid operations, checks the validity of pro, as shown in (25), (26) and (27). In addition, by deleting 'toxic waste', it can be ensured that any malicious user without certificate parameters can not forge pro for corresponding services. Hence cheating on services can be avoided. Moreover, to prevent cheating on obtaining special one-off services repeatedly, the server checks

$$\left(pk_{server}, ind, hr\right) \in \{rec\}, \tag{30}$$

where $\{rec\}$ denotes the set of service records stored in the ledger.

3) Privacy Inference Attack: In the generation scheme, the private information of the user is only available to the user himself and the APs. The APs are prevented by the embedded smart contract from disclosing the user's privacy. Also, the digest of the location certificate stored in the public ledger does not reveal the user's privacy to any malicious peer.

In the verification scheme, according to the characteristics of the trapdoor function, it is hard to get the $witness$: $\{v_{m+1}, ..., v_n\}$ from CRS and pro, so our zk-PoL only provides the necessary private information for the server without revealing any other privacy.

B. Computational Efficiency

Our experiment is on MateBook 13 with 1.8Ghz i7 CPU and 8G RAM. Moreover, the experiment is based on Circom and Snarkjs [17], [18]. Circom is a circuit generator to translate program executions to arithmetic circuits. Snarkjs is a JavaScript based cryptographic proof system for verifying the satisfiability of arithmetic circuits generated by Circom. We experiment with parameters in four levels, and the percentage of average execution time for each phase is as shown in Fig. 5.

- *Computation to QAP:* The percentages of this phase in four levels are 2.1%, 2.1%, 2.0%, and 2.0%, respectively.
- *Setup:* The percentages of this phase in four levels are 88.3%, 88.3%, 89.0%, and 88.8%, respectively.
- *Calculate witness:* The percentages of this phase in four levels all are 0.2%.
- *Generate proof:* The percentages of this phase in four levels are 9.2%, 9.2%, 8.6%, and 8.8%, respectively.
- *Verify proof:* The percentages of this phase in four levels all are 0.2%.

The result shows that the distributions of average execution time in different levels of zk-PoL are basically the same. Moreover, due to the complexity of the elliptic curve point addition and hash algorithm, the percentage of the average

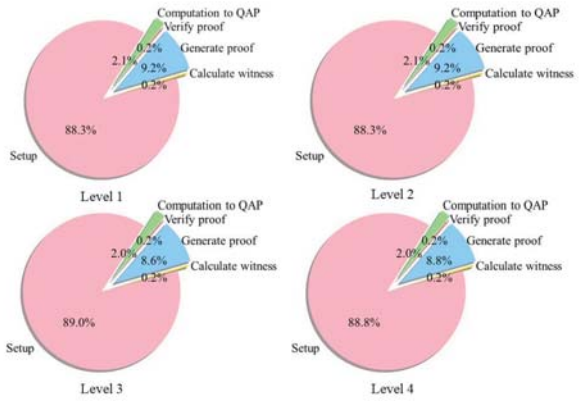

Fig. 5. Distribution of Average Execution Time

execution time of TS is large, accounting for about 90% of the total execution time. Fortunately, the TS only needs to be executed once during the system initialization. Therefore, we only need to consider the execution time of the three phases: *Calculate witness*, *Generate proof* and *Verify proof*.

The relationships between the average execution time and the number of public parameters in the three phases are shown in the Fig. 6. It can be seen that the average execution time of each phase across different public parameters is basically the same. In addition, the total execution time of the three phases that need to be repeated during different services is about 2 minutes. This execution time is acceptable in some delay-tolerant LBSs, such as the PCS case.

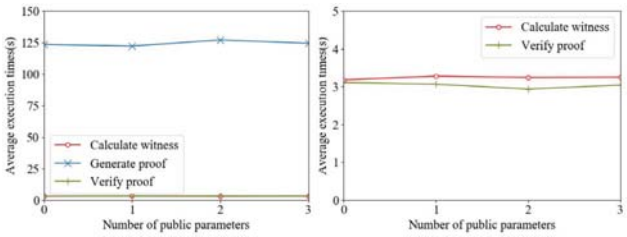

Fig. 6. Average Execution Time of Four Levels

V. Conclusion

In this paper, we bring zero-knowledge proof to LBSs for the first time, proposing zk-PoL protocol which consists of generation scheme and verification scheme. This protocol allows a user to achieve hierarchical, user-controllable privacy protection by choosing different input parameters. In addition, for main attacks, we analyze the security of our system, and the result shows that the system has excellent security against these attacks. Finally, we experiment with our system, demonstrating that the performance is independent of the privacy protection level. In diverse services, the average execution time of the phases needed to be repeated (all phases except the TS phase) is about 2 minutes, and the ratio of that to the total time cost is about 10%. This performance is appropriate to delay-tolerant LBSs, but not to delay-sensitive LBSs. In our future work, we will implement different circuit generators and cryptographic proof systems [19] to improve the system performance, making our zk-PoL appropriate to more kinds of LBSs.

References

[1] I. A. Junglas and R. T. Watson, "Location-based services," *Communications of the ACM*, vol. 51, no. 3, pp. 65–69, 2008.

[2] D. Jiang, X. Guo, Y. Gao, J. Liu, H. Li, and J. Cheng, "Locations recommendation based on check-in data from location-based social network," in *2014 22nd International Conference on Geoinformatics*. IEEE, 2014, pp. 1–4.

[3] A. K. Tyagi and N. Sreenath, "Location privacy preserving techniques for location based services over road networks," in *2015 International Conference on Communications and Signal Processing (ICCSP)*. IEEE, 2015, pp. 1319–1326.

[4] C. Javali, G. Revadigar, K. B. Rasmussen, W. Hu, and S. Jha, "I am Alice, I was in wonderland: secure location proof generation and verification protocol," in *2016 IEEE 41st conference on local computer networks (LCN)*. IEEE, 2016, pp. 477–485.

[5] Y. Li, L. Zhou, H. Zhu, and L. Sun, "Privacy-preserving location proof for securing large-scale database-driven cognitive radio networks," *IEEE Internet of Things Journal*, vol. 3, no. 4, pp. 563–571, 2015.

[6] M. Amoretti, G. Brambilla, F. Medioli, and F. Zanichelli, "Blockchain-Based Proof of Location," in *2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*. IEEE, 2018, pp. 146–153.

[7] F. Tschorsch and B. Scheuermann, "Bitcoin and beyond: A technical survey on decentralized digital currencies," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 2084–2123, 2016.

[8] S. Nakamoto *et al.*, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[9] "Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform," https://github.com/ethereum/wiki/wiki/White-Paper.

[10] M. Petkus, "Why and How zk-SNARK Works," *ArXiv*, vol. abs/1906.07221, 2019.

[11] N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer, "From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again," in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*. ACM, 2012, pp. 326–349.

[12] J. Groth, "Short pairing-based non-interactive zero-knowledge arguments," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2010, pp. 321–340.

[13] R. Gennaro, C. Gentry, B. Parno, and M. Raykova, "Quadratic span programs and succinct NIZKs without PCPs," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2013, pp. 626–645.

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

[15] 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*. IEEE, 2014, pp. 459–474.

[16] R. Shokri, G. Theodorakopoulos, J.-Y. Le Boudec, and J.-P. Hubaux, "Quantifying location privacy," in *2011 IEEE symposium on security and privacy*. IEEE, 2011, pp. 247–262.

[17] "Circom: Circuit compiler for zkSNARKs," https://github.com/iden3/circom.

[18] "Snarkjs: zkSNARK implementation in JavaScript," https://github.com/iden3/snarkjs.

[19] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza, "Succinct non-interactive zero knowledge for a von neumann architecture," in *23rd USENIX Security Symposium (USENIX Security 14)*. USENIX Association, Aug. 2014, pp. 781–796.