Hindawi
Security and Communication Networks
Volume 2017, Article ID 1076419, 9 pages
https://doi.org/10.1155/2017/1076419

Research Article

SCPR: Secure Crowdsourcing-Based Parking Reservation System

Changsheng Wan,[1,2,3] Juan Zhang,[2] and Daoli Huang[3]

[1]*School of Information Science and Engineering, Southeast University, Nanjing, Jiangsu 210096, China*
[2]*Nanjing University, Nanjing, Jiangsu 210093, China*
[3]*Key Lab of Information Network Security of Ministry of Public Security of China, Shanghai 201204, China*

Correspondence should be addressed to Changsheng Wan; wan.changsheng@163.com

Received 24 January 2017; Revised 16 April 2017; Accepted 4 May 2017; Published 28 May 2017

Academic Editor: Yacine Challal

The crowdsourcing-based parking reservation system is a new computing paradigm, where private owners can rent their parking spots out. Security is the main concern for parking reservation systems. However, current schemes cannot provide user privacy protection for drivers and have no key agreement functions, resulting in a lot of security problems. Moreover, current schemes are typically based on the time-consuming bilinear pairing and not suitable for real-time applications. To solve these security and efficiency problems, we present a novel security protocol with user privacy called SCPR. Similar to protocols of this field, SCPR can authenticate drivers involved in the parking reservation system. However, different from other well-known approaches, SCPR uses pseudonyms instead of real identities for providing user privacy protection for drivers and designs a novel pseudonym-based key agreement protocol. Finally, to reduce the time cost, SCPR designs several novel cryptographic algorithms based on the algebraic signature technique. By doing so, SCPR can satisfy a number of security requirements and enjoy high efficiency. Experimental results show SCPR is feasible for real world applications.

1. Introduction

As the amount of cars increases explosively, parking is becoming a precious resource in crowded urban areas such as New York and San Francisco [1]. To fully use parking spots that belong to "private owners (PO)," crowdsourcing-based parking reservation systems have been proposed [2, 3], where private owners can publish rental information to the "Service Provider (SP)," while other "Tenant Drivers (TD)" can download rental information from SP and make a reservation.

User privacy [4] is the basic concern for the above parking reservation system. Due to the openness of the website of SP, it is easy for malicious advertisers to download PO's private information (e.g., real identities such as user name or driver license) and annoy him by keeping on sending cheating advertisements. Moreover, during the reservation process, a terrorist may even trace the PO or the TD and establish a serious terrorist attack. Therefore, it is important to use pseudonyms instead of users' real identities in this parking reservation system, so that both cheating advertisements and terrorist attacks can be avoided. However, current crowdsourcing-based security protocols (i.e., [5–30]) are still based on real identities of PO and TD. So, to provide user privacy protection, it is urgent to develop a pseudonym-based security protocol for crowdsourcing-based parking reservation systems.

On the other hand, time cost is another serious concern for parking reservation systems. Due to the high speed of cars, the parking reservation system is a real-time application [11]. So the PO and the TD are seriously concerned about high time cost arising from running cryptographic operations. Therefore, to reduce time cost, it is desirable to use highly efficient cryptographic operations for designing security protocols for parking reservation systems. Unfortunately, current crowdsourcing-based security protocols are mainly based on the time-consuming bilinear pairing operations [5, 6]. So, to reduce time cost, it is important to develop a security protocol for crowdsourcing-based parking reservation systems without bilinear pairing.

Taking both user privacy and time cost into account, we shall design a pseudonym-based security protocol for crowdsourcing-based parking reservation systems without bilinear pairing. This security protocol should satisfy the following requirements.

(1) User Privacy. It should be guaranteed that real identities of the PO and the TD (e.g., user name or driver license) will not be extracted by an adversary. Without user privacy, the adversary may send cheating advertisements to the PO and the TD and trace them.

(2) Integrity of Rental Information. It should be guaranteed that the rental information will not be tampered by an adversary. Without integrity of rental information, the TD may download wrong rental information, resulting in parking failure.

(3) Authentication. It should be guaranteed that the PO and the TD are authenticated. Without authentication, an adversary may impersonate the TD or the PO, resulting in economic losses of the PO or the TD.

(4) Key Agreement. It should be guaranteed that the PO and the TD can negotiate a shared key for protecting subsequent transactions. Specifically, it should be guaranteed that the shared key will not be tampered or extracted by an adversary. Without key agreement, the subsequent transactions may be compromised by an adversary, resulting in reservation failure.

(5) Time Costs. It should be guaranteed that the time cost between the PO and the TD is low. Time cost is comprised of computation and communication costs. Computation cost is mainly consumed by cryptographic algorithms on the PO and the TD, while communication cost is mainly consumed by message-transmitting processes between the PO and the TD.

Obviously, designing a security protocol for crowdsourcing-based parking reservation systems is a nontrivial task, due to its complicated security and efficiency requirements as discussed above. Currently, requirement (3) has been well addressed in the literature. But, the other requirements (i.e., user privacy, integrity of rental information, key agreement, and time cost) have been largely neglected. More importantly, when considering this research topic, we find that there is no security primitive which can be directly deployed for satisfying all the above requirements. The detailed analysis for drawing this conclusion will be given in Section 2. This becomes a more urgent problem, with the deployment of parking reservation systems in real world. Motivated by this observation, we make three contributions, as described below.

(1) We discuss some security and efficiency issues in crowdsourcing-based parking reservation systems and then list a set of important requirements.

(2) We present a novel security protocol called SCPR that can fulfill all the above requirements. However, different from current crowdsourcing-based security protocols built on real identities, SCPR is based on pseudonyms. By doing so, the user privacy requirement can be fulfilled. Then, observing that the key agreement requirement is not fulfilled by current security protocols, we design a novel pseudonym-based key agreement protocol that can generate a shared key for protecting subsequent reservation transactions. Finally, observing that bilinear pairing operation is low efficient, we shall use algebraic signature [31] for designing the above security protocol. By doing so, the time cost of SCPR can be significantly reduced.

(3) We analyze the security of SCPR, showing it can fulfill requirements (1), (2), (3), and (4). And we evaluate the efficiency of SCPR, showing it can fulfill requirement (5).

The remainder of this paper is organized below. In Section 2, we discuss the related work. Then, in Section 3, we propose the SCPR protocol, followed by security analysis and efficiency evaluation in Sections 4 and 5, respectively. Finally, we draw our conclusions in Section 6.

2. Related Work

Due to its convenience, crowdsourcing has become more and more popular. For instance, [5] designed a crowdsourcing-based mobile-healthcare system. The paper [12] is a crowdsourcing-based city governance system. The paper [20] is a crowdsourcing-based system for enterprises. DYSWIS [25] introduced a crowdsourcing-based home network system. The papers [6, 21, 26] combined the cloud computing technique with crowdsourcing-based systems. The paper [27] introduced a location-based crowdsourcing scheme. The papers [18, 22] discussed the deployment of crowdsourcing technique in parking systems.

Security is the main concern for crowdsourcing-based systems. Recently, a lot of works have been focusing on this topic. For instance, [4, 7, 15, 29] analyzed the privacy-preserving and integrity of transmitted data. The paper [14] discussed the establishment of trust relationships in crowdsourcing-based systems. However, there are three issues with current works as shown below.

First, various applications may have different security requirements. For instance, in mobile-healthcare systems [5], privacy of transmitted data (personal health information) is the most important requirement, while in city governance systems [12], access control has vital significance. The security requirements of parking reservation systems are different from those of other crowdsourcing-based systems too, as discussed in Section 1. Therefore, it is desired to classify the security requirements for parking reservation systems.

Second, parking reservation systems have some special security requirements that have not been discussed by current works. For instance, in most of current protocols [5, 12, 20], real identities are used directly. However, for parking reservation systems, this may lead to a variety of security issues as discussed in Section 1. Therefore, it is desired to design a new security protocol for fulfilling those security requirements.

Third, current crowdsourcing-based parking reservation systems [18, 22] mainly focused on the data transmitting model and did not provide security protocols. For instance, [18] discussed the detailed transactions of parking reservation without designing security protocols for protecting the transactions, while [22] discussed several security issues of parking reservation systems without corresponding solutions. Therefore, it is desired to design security protocols for parking reservation systems.

At the same time, time cost is another serious concern for crowdsourcing-based systems [11, 28]. For instance, the crowdsourcing-based parking systems [18, 22] required the parking reservation protocols being executed quickly due to the high speed of cars, while the crowdsourcing-based video systems [9] required the time cost to be low due to the large amount of video data to be processed. However, current security protocols for crowdsourcing systems are mainly built on time-consuming bilinear maps. For instance, in [5, 6], both the TD and the PO have to run the pairing algorithm several times, resulting in high time cost. In Section 5, we will show that the pairing algorithm consumes much more time than other cryptographic algorithms. Therefore, it is desired to design security protocols for parking reservation systems using efficient cryptographic algorithms.

3. SCPR: The Protocol

3.1. Preliminaries. The algebraic signature [31] technique includes two processes.

The Signing Process. Given a set of secret keys $SK = \{sk_i \in Z_p, 1 \le i \le n\}$, the algebraic signature for a binary string $s = s_1 s_2 \cdots s_n$, where $s_i \in \{0, 1\}, 1 \le i \le n$, is generated as $\sigma = \sum_{i=1}^{n} s_i sk_i \bmod p$.

The Verification Process. Given a binary string $s = s_1 s_2 \cdots s_n$, where $s_i \in \{0, 1\}, 1 \le i \le n$, and the corresponding algebraic signature σ, the verifier computes $\sigma' = \sum_{i=1}^{n} s_i sk_i \bmod p$ and checks $\sigma' \stackrel{?}{=} \sigma$ to determine whether s is tampered by an attacker.

The above algebraic signature employs only several addition and modular multiplication operations. Therefore, it is highly efficient.

3.2. System Model. The system model of SCPR is shown in Figure 1, which includes three phases as illustrated below, and the notations in this paper are listed in Notations.

3.2.1. The Key-Distributing Phase. During the key-distributing phase, the SP first initializes SCPR by generating public and private system parameters. The public system parameters will be distributed to the PO and the TD, while the private system parameters will be hold by the SP. The initialization algorithm is illustrated below.

$\{SK_{SP}, PK_{SP}\} \leftarrow \text{Init}(n)$. This algorithm is run by the SP for initializing system parameters for SCPR. It takes as input the parameter of security level (i.e., n) and outputs a set of private system parameters (i.e., SK_{SP}) and the corresponding set of public system parameters (i.e., PK_{SP}).

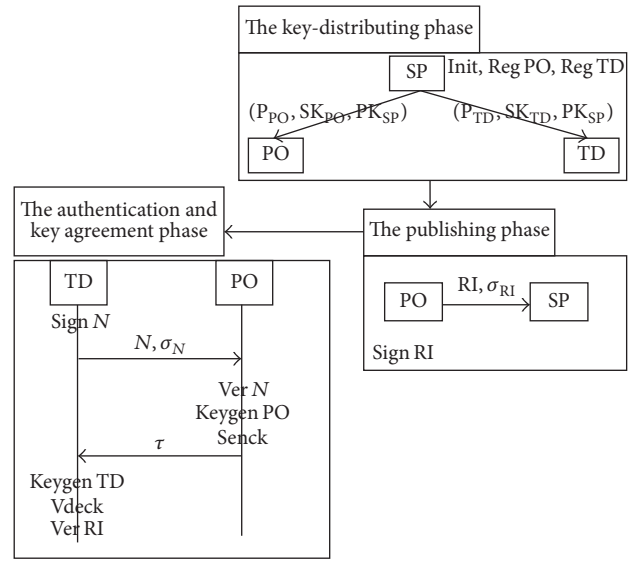

Figure 1: System model of SCPR.

Before publishing rental information, the PO sends a request to the SP. Upon receiving the request, the SP generates a pseudonym and a private key and distributes them to the PO over their preestablished secure channel, using the following RegPO algorithm.

$\{P_{PO}, SK_{PO}\} \leftarrow \text{RegPO}(RID_{PO}, SK_{SP}, PK_{SP})$. This algorithm is run by the SP for generating a pseudonym and a private key for the PO. It takes as inputs PO's real identity (i.e., RID_{PO}), the set of private system parameters (i.e., SK_{SP}), and the set of public system parameters (i.e., PK_{SP}) and outputs PO's pseudonym (i.e., P_{PO}) and a private key (i.e., SK_{PO}).

Before making a parking reservation, the TD sends a request to the SP. Upon receiving the request, the SP generates a pseudonym and a private key and distributes them to the TD over their preestablished secure channel, using the following RegTD algorithm.

$\{P_{TD}, SK_{TD}\} \leftarrow \text{RegTD}(RID_{TD}, SK_{SP}, PK_{SP})$. This algorithm is run by the SP for generating a pseudonym and a private key for the TD. It takes as inputs TD's real identity (i.e., RID_{TD}), the set of private system parameters (i.e., SK_{SP}), and the set of public system parameters (i.e., PK_{SP}) and outputs TD's pseudonym (i.e., P_{TD}) and a private key (i.e., SK_{TD}).

After the key-distributing phase, the PO holds the tuple $(P_{PO}, SK_{PO}, PK_{SP})$, and the TD holds the tuple $(P_{TD}, SK_{TD}, PK_{SP})$.

3.2.2. The Publishing Phase. When the PO wants to publish its rental information, it randomly generates a secret key (i.e., $SK_{RI} \in Z_p$) and signs the rental information using SK_{RI}. The signing algorithm is illustrated below.

$\sigma_{RI} \leftarrow \text{SignRI}(RI, SK_{RI})$. This algorithm is run by the SP for signing the rental information (i.e., RI). It takes as inputs RI and the secret key (i.e., SK_{RI}) and outputs a signature (i.e., σ_{RI}) for RI.

After the publishing phase, the PO holds SK_{RI}, and the SP gets the rental information (i.e., RI) and the signature (i.e., σ_{RI}).

3.2.3. The Authentication and Key Agreement Phase. The authentication and key agreement phase is carried out between the PO and the TD. During this phase, the TD and the PO authenticate each other and negotiate a shared key for protecting subsequent transactions. Then the PO sends SK_{RI} to the TD for checking the integrity of the rental information. The authentication and key agreement phase includes three steps as illustrated below.

Step 1. The TD establishes the authentication and key agreement phase by generating a nonce (i.e., N) and the corresponding signature (i.e., σ_N) using its own private key (i.e., SK_{TD}). Then, the TD sends N and σ_N to the PO for authentication. The signing algorithm for N is illustrated below.

$\{\sigma_N\} \leftarrow \text{Sign}N(N, P_{PO}, PK_{SP}, SK_{TD})$. This algorithm is run by the TD for generating a signature for the nonce. It takes as inputs the nonce (i.e., N), the PO's pseudonym (i.e., P_{PO}), the set of public system parameters (i.e., PK_{SP}), and the TD's private key (i.e., SK_{TD}) and outputs a signature (i.e., σ_N).

Step 2. When receiving N and σ_N from the TD, the PO checks them to authenticate the TD, using its private key (i.e., SK_{PO}) and the following Ver$N$ algorithm. If the verification succeeds, the PO generates the shared key (i.e., sk) from N using the following KeygenPO algorithm. Finally, the PO signs and encrypts SK_{RI} using sk and the following Senck algorithm and sends the signed and encrypted message to the TD. The Ver$N$, KeygenPO, and Senck algorithms are illustrated below.

$\{T, F\} \leftarrow \text{Ver}N(N, \sigma_N, SK_{PO}, P_{TD}, PK_{SP})$. This algorithm is run by the PO for authenticating the TD. It takes as inputs the nonce (i.e., N), the signature (i.e., σ_N), PO's private key (i.e., SK_{PO}), TD's pseudonym (i.e., P_{TD}), and the set of public system parameters (i.e., PK_{SP}) and outputs T if the verification succeeds, or F otherwise.

$\{\text{sk}\} \leftarrow \text{KeygenPO}(N, SK_{PO}, P_{TD}, PK_{SP})$. This algorithm is run by the PO for generating the shared key. It takes as inputs the nonce (i.e., N), PO's private key (i.e., SK_{PO}), TD's pseudonym (i.e., P_{TD}), and the set of public system parameters (i.e., PK_{SP}) and outputs the shared key (i.e., sk).

$\{\tau\} \leftarrow \text{Senck}(PK_{SP}, \text{sk}, SK_{RI})$. This algorithm is run by the PO for signing and encrypting SK_{RI}. It takes as inputs the set of public system parameters (i.e., PK_{SP}), the shared key (i.e., sk), and the publishing key (i.e., SK_{RI}) and outputs a signed and encrypted message (i.e., τ).

Step 3. Upon getting τ from the PO, the TD first generates the shared key (i.e., sk) using the following KeygenTD algorithm. Then, it decrypts and verifies τ to extract SK_{RI} and to authenticate the PO, using the following Vdeck algorithm. Finally, the TD verifies RI and σ_{RI} downloaded from the SP to make sure it is not tamped by an adversary, using SK_{RI} and the following VerRI algorithm. The KeygenTD, Vdeck, and VerRI algorithms are illustrated below.

$\{\text{sk}\} \leftarrow \text{KeygenTD}(N, SK_{TD}, P_{PO}, PK_{SP})$. This algorithm is run by the TD for generating the shared key. It takes as inputs the nonce (i.e., N), TD's private key (i.e., SK_{TD}), PO's pseudonym (i.e., P_{PO}), and the set of public system parameters (i.e., PK_{SP}) and outputs the shared key (i.e., sk).

$\{SK_{RI}, \{T, F\}\} \leftarrow \text{Vdeck}(\tau, \text{sk}, PK_{SP})$. This algorithm is run by the TD for decrypting and verifying the publishing key and authenticating PO. It takes as inputs the data to be decrypted and verified (i.e., τ), the shared key (i.e., sk), and the set of public system parameters (i.e., PK_{SP}) and outputs the publishing key (i.e., SK_{RI}). Then, it outputs T if SK_{RI} can pass the verification and the PO is authenticated, or F otherwise.

$\{T, F\} \leftarrow \text{VerRI}(\text{RI}, \sigma_{RI}, SK_{RI}, PK_{SP})$. This algorithm is run by the TD for verifying the rental information downloaded from the SP. It takes as inputs the rental information (i.e., RI), the signature (i.e., σ_{RI}), the publishing key (i.e., SK_{RI}), and the set of public system parameters (i.e., PK_{SP}) and outputs T if RI can pass the verification, or F otherwise.

After the above three phases, the PO and the TD are both authenticated and a shared key (i.e., sk) is generated for protecting the subsequent transactions between them.

From this system model, it can be seen that the PO and the TD use pseudonyms instead of their real identities. Therefore, SCPR can satisfy requirement (1) described in Section 1 (i.e., user privacy). In Section 4, we will further analyze requirement (1).

From this system model, it can be seen that the rental information (i.e., RI) is signed. Therefore, SCPR can satisfy requirement (2) described in Section 1 (i.e., integrity of rental information). In Section 4, we will further analyze requirement (2).

From this system model, it can be seen that both the PO and the TD are authenticated, and a shared key is generated between them. Therefore, SCPR can satisfy requirements (3) and (4) described in Section 1 (i.e., authentication and key agreement). In Section 4, we will further analyze requirements (3) and (4).

3.3. Construction. The construction of SCPR is a tuple (Init, RegPO, RegTD, SignRI, VerRI, Sign$N$, Ver$N$, KeygenPO, KeygenTD, Senck, and Vdeck) of probabilistic polynomial time algorithms as illustrated below.

$\{SK_{SP}, PK_{SP}\} \leftarrow \text{Init}(n)$. The SP runs this algorithm for generating system parameters for SCPR as follows. First, the SP generates a group G with a prime order p and a generator g, where n is the security level determining the key length in bit, the length of p is n-bit, and g is a randomly picked element in G. Second, the SP randomly generates a set of private keys $SK_{SP} = \{\text{skpo}_x \in Z_p, \text{sktd}_x \in Z_p, 1 \le x \le n\}$. Third, for each $\text{skpo}_x \in SK_{SP}$ and $\text{sktd}_x \in SK_{SP}$, the SP computes $\text{pkpo}_x = g^{\text{skpo}_x} \in G$ and $\text{pktd}_x = g^{\text{sktd}_x} \in G$ and gets the set of public system parameters $PK_{SP} = \{G, p, g, \text{pkpo}_x, \text{pktd}_x, 1 \le x \le n\}$.

$\{P_{PO}, SK_{PO}\} \leftarrow \text{RegPO}(RID_{PO}, SK_{SP}, PK_{SP})$. The SP runs this algorithm for generating a pseudonym and a private key for the PO as follows. First, the SP generates PO's pseudonym as $P_{PO} = h_1(RID_{PO})$, where $h_1 : Z_p \rightarrow Z_p$ is a hash function. Second, the SP computes $a_1 \cdots a_n = h_2(P_{PO})$ and takes the algebraic signature $SK_{PO} = \sum_{x=1}^{n} a_x \text{skpo}_x \bmod p$ as the PO's private key, where $h_2 : Z_p \rightarrow \{0,1\}^n$ is a hash function, and $a_1, \ldots, a_n \in \{0, 1\}$.

$\{P_{TD}, SK_{TD}\} \leftarrow \text{RegTD}(RID_{TD}, SK_{SP}, PK_{SP})$. The SP runs this algorithm for generating a pseudonym and a private key

for the TD as follows. First, the SP generates TD's pseudonym as $P_{TD} = h_1(RID_{TD})$, where $h_1 : Z_p \to Z_p$ is a hash function. Second, the SP computes $b_1 \cdots b_n = h_2(P_{TD})$ and takes the algebraic signature $SK_{TD} = \sum_{x=1}^{n} b_x sktd_x \bmod p$ as the TD's private key, where $h_2 : Z_p \to \{0,1\}^n$ is a hash function, and $b_1, \ldots, b_n \in \{0,1\}$.

$\sigma_{RI} \leftarrow SignRI(RI, SK_{RI})$. The PO runs this algorithm for generating a signature for the rental information as $\sigma_{RI} = h_1(RI \mid SK_{RI})$, where $h_1 : Z_p \to Z_p$ is a hash function.

$\{\sigma_N\} \leftarrow SignN(N, P_{PO}, PK_{SP}, SK_{TD})$. The TD runs this algorithm for generating a signature for the nonce as follows. First, the TD computes $PK_{PO} = \prod_{x=1}^{n} pkpo_x^{a_x} \in G$, where $a_1 \cdots a_n = h_2(P_{PO})$. Second, the TD computes $\sigma_N = h_1(N \mid H(PK_{PO}^{SK_{TD}}))$, where $h_2 : Z_p \to \{0,1\}^n$, $H : G \to Z_p$, and $h_1 : Z_p \to Z_p$ are hash functions.

$\{T, F\} \leftarrow VerN(N, \sigma_N, SK_{PO}, P_{TD}, PK_{SP})$. The PO runs this algorithm for authenticating the TD as follows. First, the PO computes $PK_{TD} = \prod_{x=1}^{n} pktd_x^{b_x} \in G$, where $b_1 \cdots b_n = h_2(P_{TD})$. Second, the PO computes $\sigma'_N = h_1(N \mid H(PK_{TD}^{SK_{PO}}))$ and checks $\sigma'_N \stackrel{?}{=} \sigma_N$. If this equation holds, PO returns T. Otherwise, it returns F.

$\{sk\} \leftarrow KeygenPO(N, SK_{PO}, P_{TD}, PK_{SP})$. The PO runs this algorithm for generating the shared key as follows. First, the PO computes $PK_{TD} = \prod_{x=1}^{n} pktd_x^{b_x} \in G$, where $b_1 \cdots b_n = h_2(P_{TD})$, P_{TD} is the pseudonym of TD, and $pktd_x \in PK_{SP}$ ($x \in \{1, \ldots, n\}$) are distributed from the SP during the initialization phase. Second, the PO computes $sk = PK_{TD}^{SK_{PO}N} \in G$.

$\{\tau\} \leftarrow Senck(PK_{SP}, sk, SK_{RI})$. The PO runs this algorithm for signing and encrypting SK_{RI} as follows. First, the PO encrypts SK_{RI} as $c_1 = H(sk) \oplus SK_{RI} \bmod p$, where $H : G \to Z_p$ is a hash function. Second, the PO signs SK_{RI} as $c_2 = h_1(SK_{RI} \mid H(sk))$, where $H : G \to Z_p$ and $h_1 : Z_p \to Z_p$ are hash functions. Finally, the PO gets $\tau = (c_1, c_2)$.

$\{sk\} \leftarrow KeygenTD(N, SK_{TD}, P_{PO}, PK_{SP})$. The TD runs this algorithm for generating the shared key as follows. First, the TD computes $PK_{PO} = \prod_{x=1}^{n} pkpo_x^{a_x} \in G$, where $a_1 \cdots a_n = h_2(P_{PO})$, P_{PO} is the pseudonym of PO, and $pkpo_x \in PK_{SP}$ ($x \in \{1, \ldots, n\}$) are distributed from the SP during the initialization phase. Second, the TD computes $sk = PK_{PO}^{SK_{TD}N} \in G$.

$\{SK_{RI}, \{T, F\}\} \leftarrow Vdeck(\tau, sk, PK_{SP})$. The TD runs this algorithm for decrypting and verifying the publishing key as follows. First, the TD decrypts SK_{RI} as $SK_{RI} = c_1 \oplus H(sk) \bmod p$, where $H : G \to Z_p$ is a hash function. Second, the TD computes $c'_2 = h_1(SK_{RI} \mid H(sk))$ and checks $c'_2 \stackrel{?}{=} c_2$, where $H : G \to Z_p$ and $h_1 : Z_p \to Z_p$ are hash functions. If $c'_2 = c_2$, the TD returns (SK_{RI}, T). Otherwise, it returns F.

$\{T, F\} \leftarrow VerRI(RI, \sigma_{RI}, SK_{RI}, PK_{SP})$. The TD runs this algorithm for verifying the rental information as follows. First, the TD computes $\sigma'_{RI} = h_1(RI \mid SK_{RI})$, where $h_1 : Z_p \to Z_p$ is a hash function. Second, the TD checks $\sigma'_{RI} \stackrel{?}{=} \sigma_{RI}$. If this equation holds, TD returns T. Otherwise, it returns F.

From this construction, it can be seen that SCPR does not use bilinear map. In fact, it employs only a few modular exponentiations, which is quite efficient. We will further evaluate the efficiency of SCPR in Section 5. Note that, when computing PK_{PO} and PK_{TD}, there is no modular exponentiation, because $a_x \in \{0,1\}$ and $b_x \in \{0,1\}$. So our construction is highly efficient.

4. Security Analysis

In this section, we show that SCPR can fulfill the security requirements described in Section 1 (i.e., user privacy, key agreement, integrity of rental information, and authentication).

4.1. User Privacy. In SCPR, as the PO and the TD use pseudonyms generated from their real identities such as user name or driver license, this requirement is to ensure that the adversary cannot extract real identities from pseudonyms. From Section 3.3, it can be seen that the SP generates pseudonyms using a one-way hash function (see equations $P_{PO} = h_1(RID_{PO})$ and $P_{TD} = h_1(RID_{TD})$ in the RegPO and RegTD algorithms for details). So it is straightforward that the adversary cannot extract RID_{PO} and RID_{TD} from P_{PO} and P_{TD}, and SCPR can provide user privacy protection for the PO and the TD.

4.2. Key Agreement. In Section 3.3, the PO generates sk as $sk = PK_{TD}^{SK_{PO}N} = (g^{SK_{TD}})^{SK_{PO}N} = g^{SK_{TD}SK_{PO}N}$, and the TD generates sk as $sk = PK_{PO}^{SK_{TD}N} = (g^{SK_{PO}})^{SK_{TD}N} = g^{SK_{TD}SK_{PO}N}$ (see the KeygenPO and KeygenTD algorithms in Section 3.3 for details.). So if the PO and the TD run SCPR correctly, they both can get $sk = g^{SK_{TD}SK_{PO}N}$.

Then, we show that a potential adversary cannot extract sk in three steps. In Step 1, we describe a well-known mathematical problem that will not be efficiently solved. In Step 2, we describe the adversary. In Step 3, we show this potential adversary will not be able to extract *sk* efficiently. Otherwise, we will be able to use this adversary for solving the mathematical problem. So if this mathematical problem holds, the potential adversary does not exist. Our proof is described below.

Step 1 (the (t, ϵ)-CDH problem [32]). Given $g, g^a, g^b \in G$, where $a \in Z_p$ and $b \in Z_p$ are randomly distributed unknown numbers, there is no t-time algorithm, which has the nonnegligible probability ϵ in computing $g^{ab} \in G$.

Step 2 (the adversary). In the parking reservation system, the potential adversary is between the PO and the TD. It can compute public keys of the PO and the TD from pseudonyms: $PK_{PO} = g^{SK_{PO}} = \prod_{x=1}^{n} pkpo_x^{a_x}$ and $PK_{TD} = g^{SK_{TD}} = \prod_{x=1}^{n} pktd_x^{b_x}$, where $a_1 \cdots a_n = h_2(P_{PO})$ and $b_1 \cdots b_n = h_2(P_{TD})$. Moreover, the adversary can get N.

Step 3 (the proof). If this adversary can compute $sk = g^{SK_{TD}SK_{PO}N}$ from N, $PK_{PO} = g^{SK_{PO}} = \prod_{x=1}^{n} pkpo_x^{a_x}$, and $PK_{TD} = g^{SK_{TD}} = \prod_{x=1}^{n} pktd_x^{b_x}$ with the probability ϵ in time t, we can run this potential adversary with the set of parameters $(PK_{PO} = g^a, PK_{TD} = g^b, N)$ to get $sk = g^{SK_{PO}SK_{TD}N} = g^{abN} \Rightarrow g^{ab} = sk^{N^{-1}}$. That is to say, we can use this potential adversary for solving the CDH problem [32] in time t with the probability ϵ. As the CDH problem holds, this adversary does not exist. So the adversary cannot extract sk in SCPR.

Finally, we show that a potential adversary cannot tamper sk. Since sk is generated from N, P_{PO}, and P_{TD}, to tamper sk,

the adversary has to tamper the pseudonyms or the nonce. The detailed proof can be illustrated in three steps too. Here, we just give a summarized example: If the adversary can tamper N to $M \neq N$ while still passing the Ver$N$ algorithm, it must be able to compute the signature $\sigma_M = h_1(M \mid H(\mathrm{PK}_{\mathrm{PO}}^{\mathrm{SK}_{\mathrm{TD}}})) = h_1(M \mid H(g^{\mathrm{SK}_{\mathrm{PO}}\mathrm{SK}_{\mathrm{TD}}}))$. Then, to compute σ_M, the adversary must be able to compute $g^{\mathrm{SK}_{\mathrm{PO}}\mathrm{SK}_{\mathrm{TD}}}$. As discussed above, since the adversary cannot compute $g^{\mathrm{SK}_{\mathrm{PO}}\mathrm{SK}_{\mathrm{TD}}}$ from $\mathrm{PK}_{\mathrm{PO}} = g^{\mathrm{SK}_{\mathrm{PO}}} = \prod_{x=1}^{n} \mathrm{pkpo}_x^{a_x}$ and $\mathrm{PK}_{\mathrm{TD}} = g^{\mathrm{SK}_{\mathrm{TD}}} = \prod_{x=1}^{n} \mathrm{pktd}_x^{b_x}$, it cannot tamper N to $M \neq N$.

4.3. Integrity of Rental Information. In SCPR, as the rental information is signed using the symmetric key $\mathrm{SK}_{\mathrm{RI}}$, this requirement is to ensure that the adversary cannot tamper $\mathrm{SK}_{\mathrm{RI}}$ transmitted in the authentication and key agreement phase. Moreover, from the Senck and Vdeck algorithms illustrated in Section 3.3, we can see that $\mathrm{SK}_{\mathrm{RI}}$ is signed and encrypted using the negotiated shared key sk. So the integrity of rental information is to ensure that sk is secure. Since sk is secure as discussed in Section 4.2, SCPR can provide integrity protection for rental information.

4.4. Authentication. In SCPR, as the PO and the TD communicate with each other using pseudonyms, authentication is to make sure that the pseudonyms are generated by the SP. Moreover, as $\mathrm{SK}_{\mathrm{PO}}$ and $\mathrm{SK}_{\mathrm{TD}}$ are algebraic signatures of pseudonyms (see the equations $\mathrm{SK}_{\mathrm{PO}} = \sum_{x=1}^{n} a_x \mathrm{skpo}_x \bmod p$, $\mathrm{sk}_{\mathrm{TD}} = \sum_{x=1}^{n} b_x \mathrm{sktd}_x \bmod p$, $a_1 \cdots a_n = h_2(\mathrm{P}_{\mathrm{PO}})$ and $b_1 \cdots b_n = h_2(\mathrm{P}_{\mathrm{TD}})$ in the RegPO and RegTD algorithms illustrated in Section 3.3 for details.), authentication is to prove that the PO and the TD really hold $\mathrm{SK}_{\mathrm{PO}}$ and $\mathrm{SK}_{\mathrm{TD}}$, respectively.

The proof of authentication is similar to that in Section 4.2, as illustrated by the following example. In the Sign$N$ and Ver$N$ algorithms, only when the PO and the TD hold the algebraic signatures (i.e., $\mathrm{SK}_{\mathrm{PO}}$ and $\mathrm{SK}_{\mathrm{TD}}$), they can compute $\mathrm{PK}_{\mathrm{PO}}^{\mathrm{SK}_{\mathrm{TD}}} = \mathrm{PK}_{\mathrm{TD}}^{\mathrm{SK}_{\mathrm{PO}}} = g^{\mathrm{SK}_{\mathrm{PO}}\mathrm{SK}_{\mathrm{TD}}}$. Therefore, the authentication can be reduced to the CDH problem too.

5. Efficiency Evaluation

There are many crowdsourcing-based systems (i.e., [5–30]). But, only [5, 6] aimed to design complete cryptographic algorithms and protocols [4], while other systems mainly focused on deploying crowdsourcing-based systems in multiple real world applications. So, in this section, we mainly compare SCPR with [5, 6].

Time cost is the major efficiency issue for parking reservation systems, which is comprised of computation and communication costs as discussed in Section 1. So we will first compare the computation cost of SCPR with those of [5, 6] in Section 5.1. Then, we shall compare the communication costs in Section 5.2. Finally, in Section 5.3, we will show the implementation of SCPR to make sure the newly designed protocol works well.

5.1. Comparison of Computation Costs. Computation cost is the major measure of time cost, which is mainly consumed by cryptographic algorithms. So we first tested computation costs of basic cryptographic algorithms. Then, we computed the computation costs of SCPR [5, 6].

Table 1: Basic algorithms (unit: μs).

T_{mm}	T_h	T_p	T_{me}
0.1	0.5	29540.2	577.8

T_{mm}: computation cost of modular multiplication, T_h: computation cost of hash function, T_p: computation cost of bilinear pairing, and T_{me}: computation cost of modular exponentiation.

To investigate the computation costs of basic cryptographic algorithms, we conducted the experiment on a computer with a CENTOS operating system and an Intel i7 processor. Cryptographic libraries used in this experiment include OPENSSL [33] and PBC [34]. Cryptographic group (i.e., G in Section 3) used in this experiment is the 160-bit elliptic curve [33]. Finally, we used the SHA1 hash function [33] and type F bilinear pairing parameter [34] in this experiment.

Table 1 lists the computation costs of basic cryptographic algorithms, in which the values are means of running basic cryptographic algorithms for 10,000 times. From Table 1, we can see the following.

(1) The computation costs of modular multiplication and hash function are much lower than those of modular exponentiation and bilinear pairing and can be omitted. This is because $T_h/T_p = 0.5/29540.2 \approx 1.7 \times 10^{-5}$, $T_h/T_{\mathrm{me}} = 0.5/577.8 \approx 8.7 \times 10^{-4}$, and $T_h/T_{\mathrm{mm}} = 0.5/0.1 = 5$. Therefore, in the following evaluation, we only take modular exponentiation and bilinear pairing into account.

(2) The computation cost of modular exponentiation is much lower than that of bilinear pairing. This is because $T_{\mathrm{me}}/T_p = 577.8/29540.2 \approx 2.0 \times 10^{-2}$. Therefore, by avoiding using bilinear pairing, SCPR can reduce the computation cost significantly.

Then, from Table 1, we computed the total computation costs of SCPR [5, 6]. The results are shown in Table 2 and Figure 2, where the values are total computation costs of modular exponentiation and bilinear pairing executed during the publishing phase and the authentication and key agreement phase. From Table 2 and Figure 2, we can see the following.

(1) On the PO's side, the computation costs of [5, 6] are around 10^1 to that of SCPR. This is because $(31.9n+2.9)/1.2 > (31.9 \times 1 + 2.9)/1.2 \approx 2.9 \times 10^1$ and $30.1/1.2 \approx 2.5 \times 10^1$.

(2) On the TD's side, the computation costs of [5, 6] are around $10^1 \sim 10^2$ to that of SCPR. This is because $(89.2n^2 + 59.1n)/1.2 > (89.2 + 59.1)/1.2 \approx 1.2 \times 10^2$ and $118.2/1.2 \approx 9.9 \times 10^1$.

(3) The total computation costs of [5, 6] are around 10^1 to that of SCPR. This is because $(89.2n^2+91n+2.9)/2.3 > (89.2+91+2.9)/2.3 \approx 8.0 \times 10^1$ and $148.3/2.3 \approx 6.4 \times 10^1$.

(4) On the TD's side, the computation cost of [6] will increase rapidly, when the number of attributes increases, as shown in Figure 2.

These three conclusions show that the computation cost of SCPR is much lower than those of [5, 6]. So SCPR can fulfill requirement (5) listed in Section 1.

Table 2: Comparison of computation costs (unit: ms).

	SCPR	[5]	[6]
T_{PO}	$2T_{me} = 1.2$	$T_p + T_{me} = 30.1$	$n(T_p + 4T_{me}) + 5T_{me} = 31.9n + 2.9$
T_{TD}	$2T_{me} = 1.2$	$4T_p = 118.2$	$(3T_p + T_{me})n^2 + 2nT_p = 89.2n^2 + 59.1n$
T_a	$4T_{me} = 2.3$	$5T_p + T_{me} = 148.3$	$(3T_p + T_{me})n^2 + n(3T_p + 4T_{me}) + 5T_{me} = 89.2n^2 + 91n + 2.9$

T_{PO}: computation cost of the PO, T_{TD}: computation cost of the TD, T_a: total computation cost on both the PO and the TD, and n: number of attributes in [5, 6].

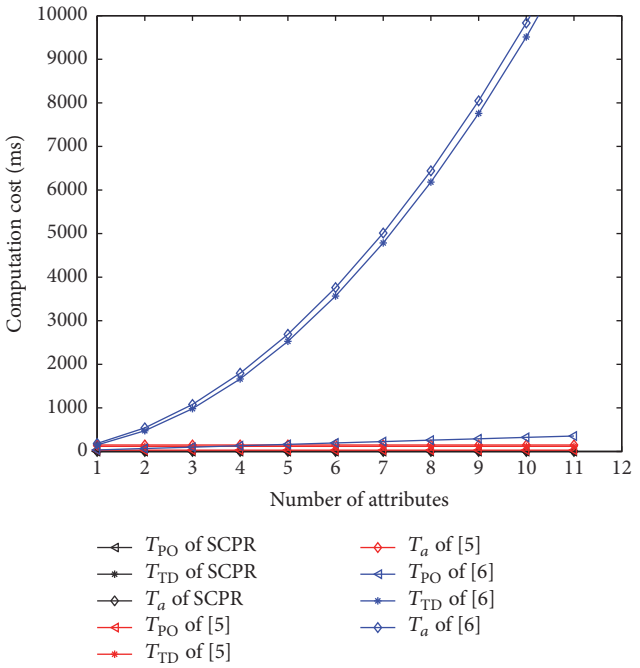

Figure 2: Evolution of computation costs.

5.2. Comparison of Communication Costs. During the publishing phase, the number of messages is one in SCPR [5, 6]. During the authentication and key agreement phase, both SCPR and [6] contain two messages, while [5] contains four messages. That is to say, SCPR and [6] contain fewer messages than [5]. So SCPR can fulfill requirement (5) listed in Section 1.

5.3. Implementation of SCPR. To make sure SCPR can work well, we implemented it. In the experiment, we use three computers, acting as the SP, the PO, and the TD, respectively. These three computers communicate with each other using 1 Gbps Ethernet. All the three computers have Intel i7 CPUs and are installed with CENTOS operating system. The cryptographic libraries used in the experiment are OPENSSL and PBC [33, 34] with the same parameters as those in Section 5.1. The Language used in our experiment is C, and the protocol for transmitting SCPR messages is TCP. Then, we get the total executing time of SCPR $\approx$ 2.4 ms. This is similar to the value computed in Table 2. Therefore, time cost is mainly consumed by cryptographic algorithms, and SCPR is feasible for real world applications.

6. Conclusion

In this paper, we have presented a security protocol for crowdsourcing-based parking reservation systems called SCPR. It can satisfy many security requirements that have not been addressed by current protocols, such as user privacy and key agreement. More importantly, to reduce the time cost, we designed several novel cryptographic algorithms for SCPR, which are quite light weight. Experimental results show SCPR is feasible for real world applications.

However, there are several more problems remaining to be solved. First, after the transaction is ended, the tenant driver and the private owner should be able to score the transaction. By doing so, the parking reservation system can provide differential services to users based on their reputation. Second, SCPR lacks a revocation method. These open issues are to be addressed in the future.

Notations

$\mathrm{RID}_{PO}, \mathrm{RID}_{TD}$: Real identities of the PO and the TD, respectively
$\mathrm{P}_{PO}, \mathrm{P}_{TD}$: Pseudonyms of the PO and the TD, respectively
$\mathrm{PK}_{SP}, \mathrm{SK}_{SP}$: Public and private system parameters of SCPR
$\mathrm{SK}_{PO}, \mathrm{SK}_{TD}$: Private keys of the PO and the TD, respectively
RI, σ_{RI}: Rental information and its signature
SK_{RI}: Secret key for signing and verifying RI
N, σ_N: Nonce and its signature
sk: The shared key negotiated between the PO and the TD
τ: The signed and encrypted data for sk
G, g, p: The cyclic group, its generator, and prime order
$h_1(), h_2(), H()$: Hash functions.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This paper is supported by the NSFC (nos. 61101088 and 71402070), the NSF of Jiangsu province (no. BK20161099), and the Opening Project of Key Lab of Information Network Security of Ministry of Public Security (no. C16604).

References

[1] D. C. Shoup, "Cruising for parking," *Transport Policy*, vol. 13, no. 6, pp. 479–486, November 2006.

[2] R. Arnott and E. Inci, "An integrated model of downtown parking and traffic congestion," *Journal of Urban Economics*, vol. 60, no. 3, pp. 418–442, 2006.

[3] J. Kincaid, "Googles open spot makes parking a breeze, assuming everyone turns into a good samaritan," http://techcrunch.com/2010/07/09/google-parking-open-spot/.

[4] K. Yang, K. Zhang, J. Ren, and X. Shen, "Security and privacy in mobile crowdsourcing networks: challenges and opportunities," *IEEE Communications Magazine*, vol. 53, no. 8, pp. 75–81, August 2015.

[5] R. Lu, X. Lin, and X. Shen, "SPOC: a secure and privacy-preserving opportunistic computing framework for mobile-healthcare emergency," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 3, pp. 614–624, March 2013.

[6] K. Yang, X. Jia, K. Ren, B. Zhang, and R. Xie, "DAC-MACS: effective data access control for multiauthority cloud storage systems," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 11, pp. 1790–1801, November 2013.

[7] D. Christin, A. Reinhardt, S. S. Kanhere, and M. Hollick, "A survey on privacy in mobile participatory sensing applications," *The Journal of Systems and Software*, vol. 84, no. 11, pp. 1928–1946, November 2011.

[8] Y. Wang, Z. Cai, G. Yin, Y. Gao, X. Tong, and G. Wu, "An incentive mechanism with privacy protection in mobile crowdsourcing systems," *Computer Networks*, vol. 102, pp. 157–171, June 2016.

[9] Ó. Figuerola Salas, V. Adzic, A. Shah, and H. Kalva, "Assessing internet video quality using crowdsourcing," in *Proceedings of the 2nd ACM international workshop on Crowdsourcing for multimedia*, pp. 23–28, the Association for Computing Machinery, Barcelona, Spain, October 2013.

[10] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM conference on Computer and communications security*, pp. 89–98, the Association for Computing Machinery, Alexandria, USA, November 2006.

[11] N. Nandan, A. Pursche, and X. Zhe, "Challenges in crowdsourcing real-time information for public transportation," in *Proceedings of the 15th IEEE International Conference on Mobile Data Management (MDM)*, pp. 67–72, IEEE, Brisbane, Australia, July 2014.

[12] G. Motta, L. You, D. Sacco, and T. Ma, "CITY FEED: a crowdsourcing system for city governance," in *Proceedings of the IEEE 8th International Symposium on Service Oriented System Engineering (SOSE)*, pp. 439–445, IEEE, Oxford, United Kingdom, April 2014.

[13] H. Zhou, J. Chen, J. Fan, Y. Du, and S. K. Das, "ConSub: incentive-based content subscribing in selfish opportunistic mobile networks," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 9, pp. 669–679, 2013.

[14] A. Tamilin, I. Carreras, E. Ssebaggala, A. Opira, and N. Conci, "Context-aware mobile crowdsourcing," in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pp. 717–720, the Association for Computing Machinery, Pittsburgh, Pennsylvania, September 2012.

[15] K. Parshotam, "Crowd computing: a literature review and definition," in *Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference*, pp. 121–130, the Association for Computing Machinery, East London, South Africa, October 2013.

[16] N. Do, C. Cheng-Hsin, and N. Venkatasubramanian, "CrowdMAC: a crowdsourcing system for mobile access," in *Proceedings of the 13th International Middleware Conference*, vol. 7662 of *Lecture Notes in Computer Science*, pp. 1–20, Springer Berlin Heidelberg, Montreal, Quebec, Canada, December 2012.

[17] E. Aubry, T. Silverston, A. Lahmadi, and O. Festor, "CrowdOut: a mobile crowdsourcing service for road safety in digital cities," in *Proceedings of he First International Workshop on Crowdsensing Methods, Techniques, and Applications*, pp. 86–91, IEEE, Budapest, Hungary, March 2014.

[18] B. Hoh, T. Yan, D. Ganesan, K. Tracton, T. Iwuchukwu, and J.-S. Lee, "CrowdPark: a crowdsourcing-based parking reservation system for mobile phones," University of Massachusetts at Amherst Tech. Report 1—14, IEEE, 2011.

[19] J. Shi, Z. Guan, C. Qiao, T. Melodia, D. Koutsonikolas, and G. Challen, "Crowdsourcing access network spectrum allocation using smartphones," in *Proceedings of the 13th ACM Workshop on Hot Topics in Networks*, pp. 1–7, ACM, Los Angeles, CA, USA, October 2014.

[20] M. Vukovic, "Crowdsourcing for Enterprises," in *Proceedings of 2009 IEEE Congress on Services (SERVICES)*, pp. 686–692, Los Angeles, CA, USA, July 2009.

[21] G. Chatzimilioudis and D. Zeinalipour-Yazti, "Crowdsourcing for Mobile Data Management," in *Proceedings of the 14th IEEE International Conference on Mobile Data Management (MDM)*, pp. 3-4, Milan, Italy, June 2013.

[22] X. Chen, E. Santos-Neto, and M. Ripeanu, "Crowdsourcing for on-street smart parking," in *Proceedings of the second ACM international symposium on Design and analysis of intelligent vehicular networks and applications*, pp. 1–8, ACM, Paphos, Cyprus, October 2012.

[23] X. Fang, J. Tang, D. Yang, and G. Xue, "Crowdsourcing to smartphones: incentive mechanism design for mobile phone sensing," in *Proceedings of the 18th annual international conference on Mobile computing and networking*, pp. 173–184, ACM, Istanbul, Turkey, August 2012.

[24] G. Chatzimilioudis, A. Konstantinidis, C. Laoudias, and D. Zeinalipour-Yazti, "Crowdsourcing with smartphones," in *Proceedings of the IEEE Internet Computing*, vol. 16, pp. 36–44, IEEE, June 2012.

[25] K. Kim, H. Nam, V. Singh, D. Song, and H. Schulzrinne, "DYSWIS: crowdsourcing a home network diagnosis," in *Proceedings of the 23rd International Conference on Computer Communication and Networks (ICCCN)*, pp. 1–10, IEEE, China, August 2014.

[26] C. L. V. Teo, *Hyrax: crowdsourcing mobile devices to develop proximity-based mobile clouds*, Carnegie Mellon University, Pittsburgh, PA, Pennsylvania, 2012.

[27] F. Alt, A. S. Shirazi, A. Schmidt, U. Kramer, and Z. Nawaz, "Location-based crowdsourcing: extending crowdsourcing to the real world," in *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries*, pp. 13–22, ACM, Reykjavik, Iceland, October 2010.

[28] I. Boutsis and V. Kalogeraki, "On task assignment for real-time reliable crowdsourcing, distributed computing systems (icdcs)," in *Proceedings of the IEEE 34th International Conference on Distributed Computing Systems (ICDCS)*, pp. 1–10, IEEE, Madrid, Spain, June 2014.

[29] A. Faggiani, E. Gregori, L. Lenzini, V. Luconi, and A. Vecchio, "Smartphone-based crowdsourcing for network monitoring:

opportunities, challenges, and a case study," *IEEE Communications Magazine*, vol. 52, no. 1, pp. 106–113, 2014.

[30] R. K. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: current state and future challenges," *IEEE Communications Magazine*, vol. 49, no. 11, pp. 32–39, November 2011.

[31] T. J. E. Schwarz and E. L. Miller, "Store, forget, and check: using algebraic signatures to check remotely administered storage," in *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems (ICDCS '06)*, p. 12, IEEE, July 2006.

[32] F. Bao, R. H. Deng, and H. Zhu, "Variations of diffie-hellman problem," in *Proceedings of the International Conference on Information and Communications Security*, vol. 2836 of *Lecture Notes in Computer Science*, pp. 301–312, Springer, Huhehaote, China, October 2003.

[33] Openssl.org, "openssl-1.0.1e.tar.gz," Feb 2013, http://www.openssl.org/source/.

[34] B. Lynn, "PBC Library Manual 0.5.11," 2006, http://crypto.stanford.edu/pbc/manual/.