This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2020.2975140, IEEE Internet of Things Journal

An Efficient, Accountable, and Privacy-Preserving Access Control Scheme for Internet of Things in A Sharing Economy Environment

Yu Liu, Kaiping Xue, *Senior Member, IEEE,* Peixuan He, David S.L. Wei, *Senior Member, IEEE,*
Mohsen Guizani, *Fellow, IEEE*

***Abstract*—Internet of things (IoT) has set off a new information technology revolution due to its convenience and efficiency. IoT enables sharing economy, as more and more people are willing to share their own things (mostly mobile devices) to leverage the under-used value. In such a situation where owners and users are often not familiar with each other, an efficient access control mechanism is needed to deal with the trust issue and support service accountability to help owners accurately get their deserved profits. Besides, in such sharing economy environment, the mobility of most shared IoT devices and their privacy preserving should also be taken into account. Regrettably, the existing schemes cannot achieve all of the aforementioned goals simultaneously and only few schemes were implemented to evaluate the claimed performance. In this paper, we propose an efficient, accountable, and privacy-preserving access control solution for IoT in a sharing economy environment. In our scheme, we utilize one-time signature to achieve anonymous authentication and let gateways store the signatures as service credentials for accountability. Meanwhile, we adopt identity-based authentication to exclude malicious gateways and shared devices from the system, and design specialized protocol for those devices moving with the users. We conduct detailed security analysis to show that our scheme can effectively defend against potential attacks, and also implement a prototype system to demonstrate that our design is indeed an efficient one.**

I. Introduction

Internet of things (IoT) has brought us into a highly connected age in recent years, in which intelligent devices and their users are connected via Internet [1]–[3] and wireless networks [4], [5]. IoT is actually changing the method of man-machine interaction and people's life style through the technologies of intelligence, automation, etc, e.g., Autonomous Driving [6], [7] and smart grid [8]–[11]. Meanwhile, *sharing economy* is developing rapidly and is bringing lots of business opportunities. Mastercard reported that only the total addressable market of shared transportation has reached 72 billion dollars and it is predicted to increase to 350 billion dollars in 2020 [12]. To adapt to the sharing economy trend, many technology companies, like Bird's shared electric skateboards [13], are sparing no effort in popularizing the shared IoT devices. It has been the trend that more and more individuals are willing to share their assets to earn some profits. Most companies or individuals provide their services using their own platforms and it brings in much trouble for users to install all kinds of platforms. So there have been some big companies, like Alibaba and Microsoft Azure, providing a united platform for all kinds of shared IoT services, and it largely improves users' experience. The combination of sharing economy and IoT has also drawn much attention from researchers in academia [14]–[16]. However, the involved security problems have been rarely investigated. Due to the higher exposure of the IoT devices in the sharing economy environment, they are much more vulnerable than those in traditional scenarios such as smart home, smart health care and so on. So security challenges for IoT in a sharing economy environment are critical issues and access control is the most fundamental one of them.

The question we need to answer is: *what is the difference between the IoT in traditional scenarios and the IoT in sharing economy environments?* The main difference is that IoT devices in traditional scenarios are not employed for profits, while in the sharing economy environment, people usually lease their IoT devices to earn profits and users must pay the owners for using the devices. Also, owners of IoT devices in traditional scenarios are barely concerned about the usage status of their devices. But owners in the sharing economy environment would like to know some feedback information (like the peak period of usage) to improve their services. Based on these observations, access control for the IoT in sharing economy environments should consider *service accountability* and *information feedback*, in addition to *device mobility*, without exposing users' privacy.

Various access control schemes have been proposed for wireless senor networks, which mainly include list-based [17] and role-based access control [18], [19] methods. But they are not scalable in IoT environment because of the *massive* number of edge devices. To better adapt them to the IoT environment, two different types of access control methods have been proposed: attribute-based encryption (ABE-based) and capability-based. In ABE-based solutions [20]–[22], to ensure the confidentiality, the data collected by the devices are encrypted by ABE before they are sent out and only authorized people or devices can decrypt the data successfully. ABE-based approaches enable fine-grained access control, but

Y. Liu is with School of Economics and Management, Hefei University, Hefei 230601, China (Email: sissi_liuyu@163.com).

K. Xue and P. He are with Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei 230027, China (Email: kpxue@ustc.edu.cn, hnythyq@mail.ustc.edu.cn).

D. Wei is with the Computer and Information Science Department, Fordham University, New York, NY 10458, USA (Email: wei@cis.fordham.edu).

M. Guizani is with the Department of Computer Science and Engineering, Qatar University, 2713 Doha, Qatar (Email: mguizani@uidaho.edu).

they bring in too much computation overhead to the resource-constrained IoT devices due to several heavy pairing related operations. In capability-based schemes [23]–[25], authorized users can get a token from a central point (e.g., cloud servers) before requesting services and they can show the granted token to IoT devices or gateways to get services. However, user identity is needed to be included in the tokens, and this information will be exposed to verifier endangering users' privacy. Besides, direct use of tokens to conduct service accounting will also leak users' privacy to IoT device providers.

The data transmitted among all kinds of IoT entities in a sharing economy environment contain a wealth of information related to user. But since the wireless links in IoT are exposed, it is effortless for attackers to get users' private information and even monitor user activities (e.g., when users use sharing bikes to go to work and which path they chose to take) by eavesdropping [26]. So privacy protection should be taken into account and well addressed. A few solutions have been proposed to preserve users' privacy [17], [27]–[29]. Unfortunately, these proposed schemes make it hard for IoT device providers to get useful feedback information. So these solutions cannot meet the security demands in the sharing economy environment.

Motivated by these observations, in this paper we propose an efficient, accountable, and privacy-preserving access control for IoT in the sharing economy environment. In our scheme, we utilize identity-based authentication to make gateways only discover services provided by legitimate IoT devices. We also further make decentralized gateways authenticate users directly through one-time signatures [30] generated by users to keep anonymous rather than leveraging a central server. Even when the server breaks down, our system can still work properly. Moreover, for service accounting, these signatures can be used as trusted service credentials, but it is irrational for the central server to verify each single signature due to its huge overhead. Therefore, we make gateways aggregate collected signatures regularly and the central server can only verify the aggregated signatures to check the validity of the credentials received from gateways. Our contributions can be summarized as follows:

- ○ We propose a secure and efficient access control scheme for IoT in sharing economy environments. Our scheme can support mobility and service accountability, and the operations in the processes would not affect users' experience much.
- ○ We introduce one-time signature to accomplish anonymous authentication and make services accountable by aggregating one-time signatures. Besides, IoT device providers improve their services by collecting other feedback information, provided they cannot get any user's privacy information.
- ○ We thoroughly analyze the security strength of our scheme and implement a prototype system to evaluate the performance of main phases in our system.

The rest of this paper is organized as follows. Section II reviews the related work. Section III describes our system model, security assumptions, design goals, and preliminaries, while Section IV presents the details of our proposed scheme. Section V and Section VI show the security analysis and performance evaluation, respectively. Finally, in Section VII we conclude our work.

II. Related Work

Since the beginning of the 21th century, access control has caused widespread concern in the field of wireless sensor networks. The intuitive thought is maintaining an access control list (ACL) at the sensors on the owner's side, similar to the work in [17] proposed by He *et al.*, to decide who can access a certain sensor. Afterwards, to ease the privilege management in ACL-based schemes, role-based access control schemes were proposed [18], [19]. In these schemes, the sensor owner assigns privileges to a role instead of a person, which is more efficient. However, due to the huge number of the IoT devices, managing the privileges becomes more intractable, so these methods can not be used in IoT directly.

To better solve the access control problem in IoT, many new methods have been proposed. Attribute-based encryption is one of the popular methods widely used in many network sceneries [31]–[34], including IoT. Phuong *et al.* [20] proposed puncturable attribute-based encryption to make sure that the sender can revoke the compromised IoT devices' decryption capability for the past messages in time. Zhang *et al.* [21] proposed to hide some sensitive attributes in access policies of CP-ABE to protect privacy and add a decryption test to improve the decryption efficiency. As shown in [22], the direct use of ABE in IoT indeed brings in much computation overhead because ABE needs to conduct heavy pairing related operations for several times. Therefore, it is unfriendly and unadaptable to resource-constrained IoT devices. Besides, the problem to reduce the computation overhead in IoT devices without increasing much communication overhead is not well dealt with in these schemes.

The capability-based access control is another popular type of methods because of its flexibility that can meet the various requirements of different IoT architectures. This type of schemes uses a central point (e.g., backend in [23], owner in [25], and specialized server in [24]) to authenticate users and assign tokens to users. Users request services with the received tokens, and IoT devices or gateways verify the tokens to decide whether to provide services. However, since the tokens in these schemes always contain private information (like user identity, user privileges, etc.), and verifier can easily get these private information, so these schemes cannot satisfy the security demands considering privacy protection.

In the IoT scenario, the data are produced when the users are using IoT devices and always contain users' behavior, identity and some other critical private information. Meanwhile, because of the exposed and dynamic environment, these private information is easily compromised by malicious attackers [35]. Thus, privacy preserving in such an IoT environment has become a vital issue [17], [27]–[29], [36]. Some schemes try to design privacy-preserving protocol for IoT using temporary identity [27], hash function [29] or ring signature [17]. Aitzhan *et al.* [28] proposed a solution to ensure the privacy preserving via multi-signatures and blockchain. However, these schemes are proposed for specific IoT applications like smart home and smart healthcare, which is greatly different from IoT in a sharing economy environment. Specifically, they are incapable of service accountability, information feedback and device mobility support. Differential privacy and homomorphic encryption are also two important cryptographic techniques to provide privacy preserving in many network scenarios [8], [9], [37], [38], which are mainly used for privacy protection of private data. Some other privacy enhancing techniques, e.g., trusted execution environment, are also leveraged to achieve data's privacy preservation [10], [39], [40].

III. Model, Assumption, Goals, and Preliminaries

A. System Model

1) Components: Our system model is similar to the model proposed in smart homes [27], [41] with minor modification. As shown in Fig. 1, our system is mainly composed of users, gateways, a central server, and IoT device providers with their shared IoT devices.

Users connect with gateways to obtain IoT services through the subject devices (e.g., smartphones). We assume that the subject devices have a considerable degree of computation and storage capability (e.g., 2.3GHz CPU and 64GB ROM).

Gateways are connected with a large number of shared IoT devices. They are responsible for authenticating users through the signatures received from users, aggregating the signatures as the trusted credentials, and sending commands to the connected devices. Note that at first IoT device providers can pay some institutions at first for gateway provision to run the fundamental services. In order to improve the service, individuals are also allowed to provide gateways for helping shared IoT services subsequently. For increasing the motivation of providing gateways, IoT device providers need to give individuals some incentive in economic according to the number of signatures that the gateways have helped authenticate. In such a way, gateways provided by institutions and individuals can cover a large area to enable users to enjoy the services whenever and wherever they want to. We assume the gateways have constrained computation capability but sufficient storage capability (e.g., 1.2GHz CPU and tens of GB ROM).

A central server is a united platform for all of the IoT device providers like Alibaba, and it is responsible for key management and fund management. Users can buy tokens from the central server to obtain IoT services and IoT device providers can get their profits from it according to the number/time of services their devices provide. Note that, there are two kinds of IoT devices according to their charge ways: pay-per-unit-time (e.g., shared smart cars) and pay-per-use (e.g., shared smart printers). For simplicity, we call pay-per-unit-time shared IoT devices *type A devices* and pay-per-use shared IoT devices *type B devices* in this paper. Besides, IoT device providers provide shared IoT devices. These devices are often resource-constrained with weaker computation capability (e.g., 1.2GHz CPU or weaker).

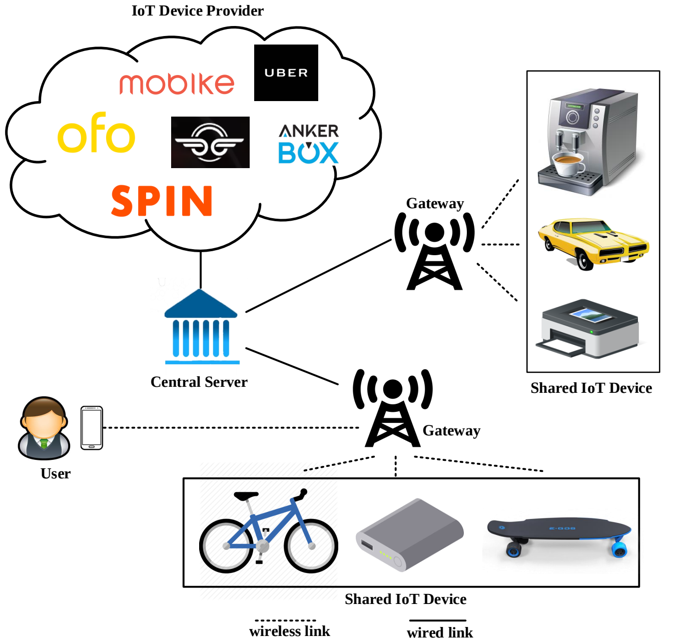


Fig. 1. System Model

2) Communication Model: Users' subject devices and shared IoT devices can communicate with gateways through wireless links in many ways (e.g., WiFi, Bluetooth, ZigBee, etc.) and they can also connect with a central server by WiFi, GPRS, etc. Gateways and IoT device providers communicate with the central server through wired links (e.g., Ethernet). Besides, we assume there exist secure channels between gateways/shared IoT devices and the central server.

B. Security Assumption

We assume that users are untrusted in our system and they will try to pay as little money as possible to get as much services. The IoT device providers and their shared IoT devices are assumed to be rational but greedy. On the one hand, big IoT device companies (e.g., ofo, SPIN) will not risk providing malicious IoT devices that cannot work to maintain their good reputations. On the other hand, some individuals may provide malicious devices to get some profits.

Unlike traditional IoT environment, in this paper, we assume that gateways are semi-trusted. They will follow the pre-designated protocol faithfully. But to earn more profits, they may claim more services they provided by forging more signatures which are the accounting credentials. Besides, they may be curious about the privacy of users, e.g., who prefers which kind of coffee, etc. Finally, the central server is assumed to be trusted and it can be a committee composed of some big IoT device providers in a real-world scenario.

C. Design Goals

In this paper, we would like to design an efficient and secure access control scheme for IoT in sharing economy environments with the following goals:

- ◦ ***Secure access control.*** The proposed scheme should conduct access control accurately and block unauthorized users outside of the system. Also, malicious gateways and devices cannot join the system.

- ***Privacy preservation.*** Our solution need to ensure that malicious attackers are unable to infer any users' privacy information, such as user identity, user preference, user moving track, etc. While preserving privacy, shared IoT device providers can get the information which can be used to improve their services normally.
- ***Efficient service accountability.*** Our proposed approach should provide service accounting mechanism, in which central server can efficiently know the exact amount of services offered by IoT device providers so that providers and gateways are able to get their profits accurately.
- ***Mobility Support.*** Our scheme is required to address the trust and accounting problem brought by the situation where shared IoT device moves with users in the sharing economy environment.

D. Preliminaries

From the security perspective, the security of our scheme is based on the intractability of discrete logarithm problem (DLP), Computation Diffie-Hellman (CDH) assumption and Divisible Computation Diffe-Hellman (DCDH) assumption on the multiplicative cyclic group G_1 [42].

Definition 1: **DLP.** The DLP is, given $g, h = g^x \in G_1$, to compute $x = log_g h$.

Definition 2: **CDH assumption.** Given $(g^a, g^b \in G_1)$ for unknown $a, b \in Z_q^*$, it is infeasible to compute g^{ab}.

Definition 3: **DCDH assumption.** Given $(g^a, g^b \in G_1)$ for unknown $a, b \in Z_q^*$, it is infeasible to compute $g^{a/b}$.

IV. Proposed Scheme

A. System Overview

Fig. 2 shows the overview of our system. As shown in this figure, in our system, entities including users, gateways and shared IoT devices are required to register to central server for getting corresponding secret keys. Since in a sharing economy environment, all these three entities cannot be fully trusted, thus before communicating with other entities, they must authenticate the communication peers first. Specifically, in service discovery phase, we utilize identity-based mutual authentication protocol to keep malicious gateways and shared IoT devices outside of the system. Besides in service request phase, we make gateways and users show identity-based signature (IBS) and one-time signature (OTS), respectively, to identify themselves.

Moreover, we design a special protocol (i.e., service termination) for the situation that shared IoT devices move with users to a new place. To record the exact amount of one-time signatures authenticated by gateways, in service accounting phase, gateways aggregate the signatures they collect regularly, and it is easy for central server to conduct service accounting by verifying the aggregated signatures. To let central server improve its services in time without exposing users' privacy, gateways send device related information to central server regularly. Then, to make our system more robust, the centralized point (e.g., a central server) is not involved in authentication process to users.

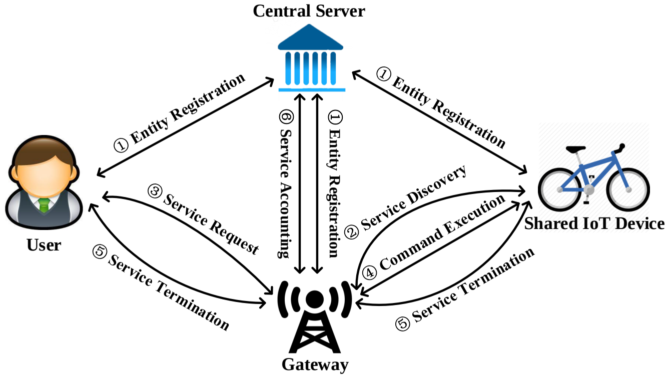


Fig. 2. System Overview

Next we will describe the details of our system in eight phases: system initialization, entity registration, service discovery, service request, command execution, service termination, service accounting, and entity revocation.

B. System Initialization

In this step, central server initializes the system and generates the public and private parameters as follows:

- Generate a bilinear map group system $S = (q, G_1, G_T, e(.,.))$., where G_1, G_T are multiplicative cyclic groups of the same order q. Randomly select generators $g_1, h \in G_1$ and set $g_2 = e(g_1, g_1)$.
- Select a random master private key $s \in Z_q^*$ and compute the corresponding public key $\lambda = {g_1}^s$.
- Choose several cryptographic hash functions $H_1 : \{0,1\}^* \rightarrow \{0,1\}^*$, $H_2 : \{0,1\}^* \rightarrow Z_q^*$.
- Publish the system public parameters as: $(S, g_1, g_2, h, \lambda, H_1, H_2, E(.))$, where $E(.)$ is a symmetric encryption algorithm.

C. Entity Registration

In this step, gateway i (with its identity ID_i) and shared IoT device j (with its identity ID_j) need to send their identities to the central server for registration. For gateways, the central server computes:

$$PK_i = H_2(ID_i||TS_i), SK_i = {g_1}^{1/(s+H_2(ID_i||TS_i))},$$

where TS_i is the current timestamp. Then the central server randomly selects $r_i \in Z_q^*$ and computes:

$$R_i = {g_1}^{r_i}, A_i = r_i + sH_2(ID_i||TS_i||R_i).$$

Finally, gateway i can get its public keys PK_i and private keys (SK_i, R_i, A_i) from the central server.

For each shared IoT device j, the central server computes:

$$PK_j = H_2(ID_j||TS_j), SK_j = {g_1}^{1/(s+H_2(ID_j||TS_j))}.$$

and generates a signed *profile*$_j$, which states the provided services of IoT device j, the provider to which the devices belongs, etc. Then the central server sends the public key

PK_j, the private key SK_j, and the signed *profile*$_j$ to the IoT device. To be noted, as the existence of the timestamp TS_j, which represents the time when implementing key generation, the central server should periodically update secret keys for legitimate gateways and shared IoT devices. It's outside the scope of this paper, so we won't cover the details of this.

In the system we use one-time signatures as accounting credentials. Secret keys associated with one-time signatures can be used only once, and the central server must generate massive secret keys in advance. To manage the massive secret keys, the central server assigns a l-bit identifer pid to each pair of secret keys and utilizes a bitmap to record the unused pids. Suppose there are several charge choices (e.g., 1 dollar, 5 dollars, and 10 dollars). Users can pay money for requesting secret keys according to the charge choices with a certain exchange ratio (e.g., 1 dollar for requesting 5 keys). The central server can generate secret keys for different charge choices in advance. First, it generates an original key pair (UPK_0, USK_0) as follows:

$$USK_0 = (b_i \xleftarrow{R} Z_q^*, c_i \xleftarrow{R} \{0,1\}^{l_r})_{i=1}^m,$$

$$UPK_0 = (v_i \leftarrow {g_1}^{b_i} h^{c_i})_{i=1}^m,$$

where l_r represents the length of c_i, and m is the number of b_i/c_i pairs, which is related to signature generation for allowed length of messages. Then the central server further generates a set of keys USK_i for $i \in [1, m]$, as shown in Fig. 3, we can get:

$$b_{i,\kappa} = H_2(b_{i,\kappa-1}),\ \ c_{i,\kappa} = H_1(c_{i,\kappa-1}),\ \ \kappa \in [2, n], \qquad (1)$$

where $b_{i,1} = H_2(b_i)$ and $c_{i,1} = H_1(c_i)$. Then it computes all the corresponding UPK_i and generates an unused pid (pid_j) to associate with each UPK_i/USK_i pair. To be noted that, the parameter n is determined by charge choice and exchange ratio. For example, if the charge choice is 5 dollars and exchange ratio is 1 dollar for 5 keys, n is 24.

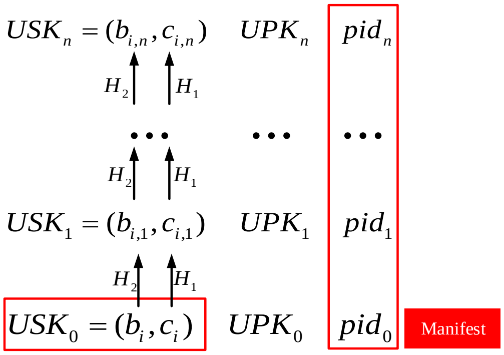


Fig. 3. Key Generation

The central server only stores USK_0 and all the corresponding pids (e.g., pid_0, pid_2,$\cdots$, pid_n), called *manifest*, for reducing storage overhead. In order to access IoT devices, when user k pays some money to request a specific number of secret keys, the central sever will select an appropriate *manifest* from the storage and send it to user k. After receiving the manifest, the user can recover the remaining secret keys as Eq. 1. Besides, the central server needs to send all the records $< pid, UPK >$ to the legitimate gateways and update these records every day.

D. Service Discovery

To join the system, a shared IoT device sends its authentication request to the nearby gateway to register their services, and then a mutual authentication will be conducted between the IoT device and the gateway, which is shown in Fig. 4. In this paper, we utilize an identity-based mutual authentication protocol, like the one in [43], to conduct the mutual authentication.

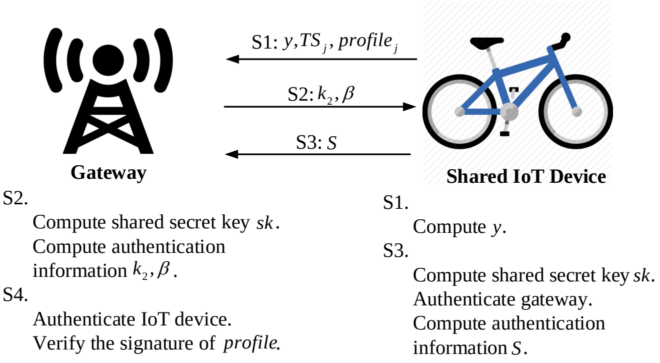

S1: y, TS_j, $profile_j$
S2: k_2, β
S3: S
Gateway
Shared IoT Device
S2.
Compute shared secret key sk.
Compute authentication information k_2, β.
S4.
Authenticate IoT device.
Verify the signature of *profile*.
S1.
Compute y.
S3.
Compute shared secret key sk.
Authenticate gateway.
Compute authentication information S.

Fig. 4. Service Discovery Process

The details of communications between the shared IoT device j and gateway i are as follows:

S1: IoT device j → **Gateway** i**:** $\{y, TS_j, \textit{profile}_j\}$.
Shared IoT device j randomly chooses $\alpha \in Z_q^*$ and computes $x_1 = H_2(\alpha||SK_j)$. Then it computes $y = ({g_1}^{H_2(ID_i||TS_j)}\lambda)^{x_1}$, and then sends y and $profile_j$ to gateway i.

S2: Gateway i → **IoT device** j**:** $\{k_2, \beta\}$.
Gateway i chooses a random number $\delta \in Z_q^*$ and computes $x_2 = H_2(\delta||SK_i)$. Then it generates the secret key sk by computing:

$$\begin{aligned} k_1 &= e(y, SK_i) = {g_2}^{x_1}, \\ sk &= H_2({k_1}^{x_2}) = H_2({g_2}^{x_1 x_2}). \end{aligned} \qquad (2)$$

Afterwards, the gateway computes $k_2 = g_2^{x_2}, \beta = H_1(sk||k_1||k_2||ID_i||y)$ and sends them to IoT device j to make it generate the secret key and authenticate the gateway.

S3: IoT device j → **Gateway** i**:** $\{S\}$.
IoT device j obtains the shared secret key by computing $sk = H_2({k_2}^{x_1}) = H_2({g_2}^{x_1 x_2})$. Then it computes $k_1 = {g_2}^{x_1}, \beta' = H_1(sk||k_1||k_2||ID_i||y)$ and authenticates the gateway by checking whether $\beta' = \beta$. Finally it computes $S = {SK_j}^{x_1+sk}$, and then sends it to the gateway and stores $< ID_i, sk >$.

S4: Gateway i checks whether $e(S, {g_1}^{H_2(ID_j||TS_j)}\lambda) = k_1 \times {g_2}^{sk}$. Then the gateway verifies the signature of *profile*$_j$ and store the related information as $< ID_j, sk, \textit{profile}_j >$.

E. Service Request

As shown in the left side of Fig. 5, in the request phase, users send one-time signatures to gateways to show that they are the authorized users who have paid for the services. But before users sending signatures, they need to authenticate gateways to prevent malicious gateways stealing their signatures without helping them get corresponding services. Here we use an identity-based signature to let gateways show their legitimate identities. The request process is as follows:

S1: **User** k $\rightarrow$ **Gateway** i**:** $\{Squery, \eta\}$.
User k selects a random number $\eta \in Z_q^*$ and sends service query message and the random number η to the gateway.

S2: **Gateway** i $\rightarrow$ **User** k**:** $\{\sigma, \theta, SL\}$.
The gateway selects two random numbers $d \in Z_q^*$ and $\theta \in \{0,1\}^{n'}$ with n' satisfying $C_m^{\lfloor m/2 \rfloor} > 2^{n'}$. Then it computes $D_i = {g_1}^d, z = d + A_i \times H_2(ID_i||TS_2||\eta||\theta||R_i||D_i)$ and gets the identity-based signature $\sigma =< R_i, D_i, z, TS_2 >$. Finally, the gateway sends σ, θ, SL to the user, where SL is the available services list consisting of shared IoT devices' identities, the type of the services, and the providers to which they belong, etc.

S3: **User** k $\rightarrow$ **Gateway** i**:** $\{E_{sk'}(cmd, parm, ID_j, pid)$, σ',$R'\}$.
The user verifies the signature σ by checking $g_1^z = D_i \times (R_i \times \lambda^{H_2(ID_i||TS_1||R_i)})^{H_2(ID_i||TS_2||\eta||\theta||R_i||D_i)}$. If the verification passes, it is reasonable for the user to believe that the gateway is legitimate and it stores the tuple (σ, θ, η). Then he/she can generate a one-time signature by using **Algorithm** 1. Afterwards, the user selects a random number $r' \in Z_q^*$ and computes $R' = g_1^{r'}, sk' = D^{r'}$. Finally, he/she chooses the service according to SL and sends the encrypted command. The cmd in it is the command that the user wants to execute, like turn on the machine, which can be further clarified by the parameter $parm$. For example, when the user wants to get shared coffee machine service, the $parm$ here can be the type of coffee he/she would like to drink.

```
Algorithm 1: One-time Signature Generation
Input: Received random number $\theta$, the security
       parameter $n$, unused secret key $USK = (b_i, c_i)$.
Output: A valid signature.
$temp = ka = \lfloor m/2 \rfloor$;
for i=1 to m do
    if $\theta > C_{n-i}^{ka}$ then
        $\epsilon_{temp-ka+1} = b_i$, $\rho_{temp-ka+1} = c_i$;
        $\theta = \theta - C_{n-i}^{ka}$, $ka = ka - 1$;
    end
end
$\epsilon = \epsilon_1||\epsilon_2||\dots||\epsilon_{temp}$;
$\rho = \rho_1||\rho_2||\dots||\rho_{temp}$;
return $\sigma' = (\epsilon, \rho)$
```

S4: Gateway i computes $sk' = R'^d$ and decrypts the encrypted command. Then it verifies the signature by using **Algorithm** 2 and stores $(\epsilon_{\text{total}}, \rho_{\text{total}}, pid, \theta)$ for future accounting. Meanwhile, it maintains a counting table to record the number of signatures received for different shared IoT providers and it increases the number by 1 for the corresponding provider.

```
Algorithm 2: One-time Signature Verification
Input: The random number $\theta$, the security parameter
       $m, l_r$, the secret key $UPK$ with identity $pid$.
Output: Valid or Invalid.
$temp = ka = \lfloor m/2 \rfloor$;
$\epsilon_{\text{total}} = \sum_{i=1}^{ka} \epsilon_i$, $\rho_{\text{total}} = \sum_{i=1}^{ka} \rho_i$;
if $0 \leq \rho_{\text{total}} \leq m(2^{l_r} - 1)/2$ then
    $temp_1 = g_1^{\epsilon_{\text{total}}} h^{\rho_{\text{total}}}, temp_2 = 1$;
    for i=1 to m do
        if $\theta > C_{n-i}^{ka}$ then
            $temp_2 = temp_2 \times v_{temp-i+1}$;
            $\theta = \theta - C_{n-i}^{ka}$, $ka = ka - 1$;
        end
    end
    if $temp_1 == temp_2$ then
        return Valid;
    end
end
return Invalid;
```

F. Command Execution

The right side of Fig. 5 shows the process of command execution. In this phase, the gateway will help the user get the service by encrypting the command through the secret key shared with the shared IoT device. The gateway finds the shared secret key sk with device i and it communicates with the device as follows:

S1: **Gateway** i $\rightarrow$ **Device** j**:** $E_{sk}(cmd, parm, TS)$.
Gateway i generates timestamp TS and implements Encryption over message $\{cmd, parm, TS\}$. Then send the encrypted message to device j.

S2: **Device** j $\rightarrow$ **Gateway** i**:** $E_{sk}(fbac, TS)$.
After the device received the message, it first decrypts the message and verifies whether timestamp TS_3 is in a permitted time period. Then it executes the corresponding service and returns some feedback information $(E_{sk}(fbac, TS))$, where $fbac$ can be the current status feedback information, such as normal, damaged, the remaining power, etc.

Finally, the gateway sends the feedback information collected to the corresponding IoT device providers regularly. This can help IoT device providers know the status of their devices, the peak period of usage, and the popular locations where the services are provided so that providers can improve their services in time according to these information.

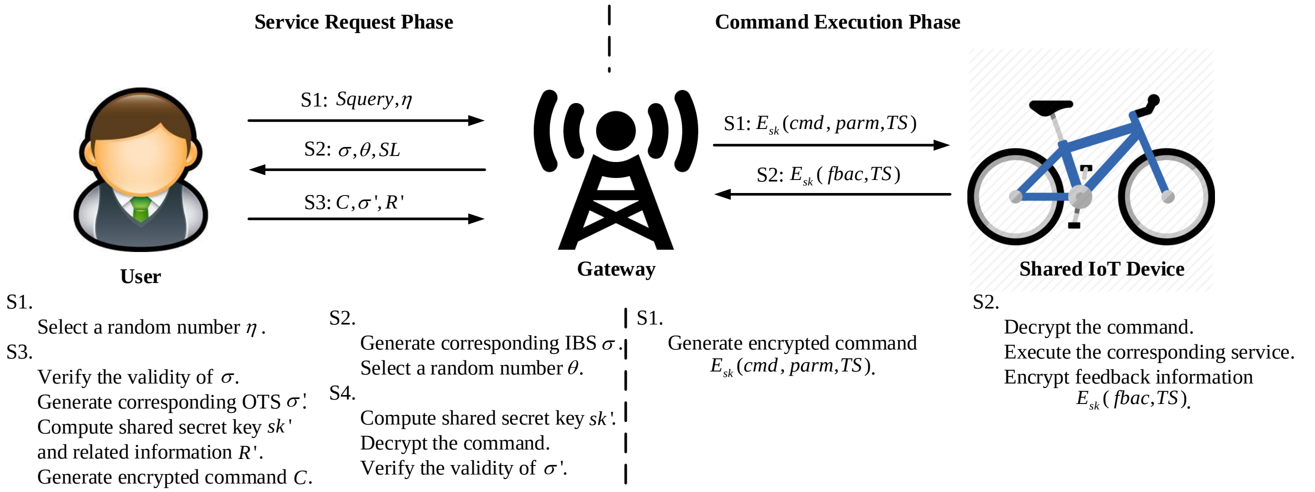

Service Request Phase
Command Execution Phase
S1: $Squery, \eta$
S2: σ, θ, SL
S3: C, σ', R'
S1: $E_{sk}(cmd, parm, TS)$
S2: $E_{sk}(fbac, TS)$
User
Gateway
Shared IoT Device
S1.
Select a random number η.
S3.
Verify the validity of σ.
Generate corresponding OTS σ'.
Compute shared secret key sk'
and related information R'.
Generate encrypted command C.
S2.
Generate corresponding IBS σ.
Select a random number θ.
S4.
Compute shared secret key sk'.
Decrypt the command.
Verify the validity of σ'.
S1.
Generate encrypted command
$E_{sk}(cmd, parm, TS)$.
S2.
Decrypt the command.
Execute the corresponding service.
Encrypt feedback information
$E_{sk}(fbac, TS)$.

Fig. 5. Service Request and Command Execution Process

G. Service Termination

In traditional IoT scenario, the whole process will end when the command execution is finished. However, in a sharing economy environment, most IoT devices move with the users and the users may return them back at a new place. Therefore, an additional phase service termination is proposed to deal with the mobility.

1) For *type B devices*, it will be terminated automatically once it finishes the service (e.g., a shared smart printer finishes printing a paper).

2) For *type A devices*, we design a service termination protocol, which is illustrated by Fig. 6. In the protocol, the user needs to terminate the device on his/her own initiative. When the user moves to a new place connecting with a new gateway to return the devices back, the communication process to be performed is as follows:

S1: User k → New Gateway i': $Rtn, (\sigma, \theta, \eta), ID_i, \eta'$.
Here η' is a random number in Z_q^*, ID_i is the identity of the original gateway, (σ, θ, η) is the tuple that the user stored in service request phase.

S2: Gateway i' → User k: $\sigma_{new}, num, (\theta_1, \dots, \theta_{num})$.
New gateway i' first verifies the the signature σ. Then the new gateway computes the service time that the user enjoys the service $\Delta T = TS_3 - TS_2$, where TS_3 is the current timestamp and TS_2 is the timestamp in σ, and num is the number of one-time signatures the user needs to give according to the charging standard. The new gateway generates a signature σ_{new}, like *request* phase using η', and selects num random numbers $(\theta_1, \dots, \theta_{num})$.

S3: User → Gateway i': $R'_{new}, (\sigma'_1, \dots, \sigma'_{num}), \sigma', E_{sk'_{new}}($ $ID_{j'}, cmd, parm, pid_1, \dots, pid_{num})$.
The user verifies σ_{new} and generates corresponding number of one-time signatures using $(\theta_1, \dots, \theta_{num})$. Then he/she generates the secret key, like step 4 in service request phase using R'_{new}, and sends the one-time signature generated in service request phase σ', new num one-time signatures, and encrypted termination command to the gateway.

S4: After receiving the message, the new gateway first verifies σ' using θ, and then it verifies other received signatures. If the verification passes, it sends a successful message to the user.

Then the gateway conducts the mutual authentication with device j' like the steps in the service discovery phase and sends encrypted termination command message to the device. $S5 \sim S9$ in Fig. 6 are the detailed statements of these processes. Besides, it records $(ID_i, ID_{j'}, ID_{i'}, \Delta T)$ and sends them to IoT providers regularly. So that the providers can get the information about the popular services and locations where users return IoT devices back.

Algorithm 3: Aggregated Signature Verification

```
Input: The random numbers $\theta_1, \dots, \theta_n$, the security
       parameter $m$, the secret keys $UPK_1, \dots, UPK_n$
       with identities $pid_1, \dots, pid_n$, the aggregated
       signature $(\epsilon_a, \rho_a)$.
Output: Valid or Invalid.
for $i$=1 to $n$ do
    $temp = ka = \lfloor m/2 \rfloor, temp_1 = 1$;
    for $j$=1 to $m$ do
        if $\theta_i > C_{n-i}^{ka}$ then
            $temp_1 = temp_1 \times v_{temp-j+1}$;
            $\theta_i = \theta_i - C_{n-i}^{ka}$, $ka = ka - 1$;
        end
    end
end
$temp_2 = g_1^{\epsilon_a} h^{\rho_a}$;
if $temp_1 == temp_2$ then
    return Valid;
end
return Invalid;
```

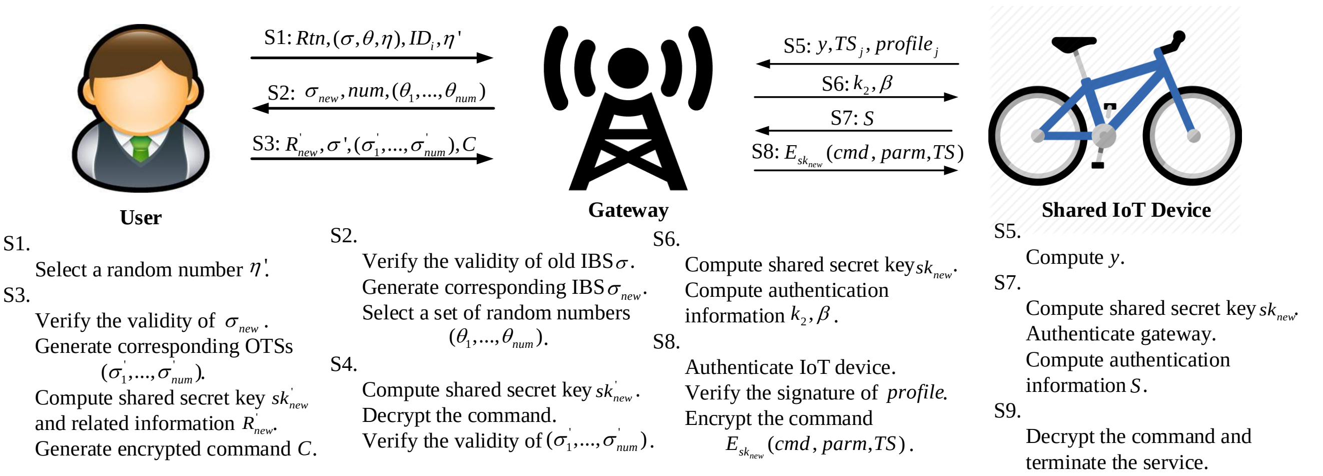

S1: $Rtn,(\sigma,\theta,\eta),ID_i,\eta'$
S2: $\sigma_{new},num,(\theta_1,...,\theta_{num})$
S3: $R'_{new},\sigma',(\sigma'_1,...,\sigma'_{num}),C$
S5: $y,TS_j,profile_j$
S6: k_2,β
S7: S
S8: $E_{sk_{new}}(cmd,parm,TS)$
User
Gateway
Shared IoT Device
S1.
Select a random number η'.
S3.
Verify the validity of σ_{new}.
Generate corresponding OTSs $(\sigma'_1,...,\sigma'_{num})$.
Compute shared secret key sk'_{new} and related information R'_{new}.
Generate encrypted command C.
S2.
Verify the validity of old IBS σ.
Generate corresponding IBS σ_{new}.
Select a set of random numbers $(\theta_1,...,\theta_{num})$.
S4.
Compute shared secret key sk'_{new}.
Decrypt the command.
Verify the validity of $(\sigma'_1,...,\sigma'_{num})$.
S6.
Compute shared secret key sk_{new}.
Compute authentication information k_2,β.
S8.
Authenticate IoT device.
Verify the signature of $profile$.
Encrypt the command $E_{sk_{new}}(cmd,parm,TS)$.
S5.
Compute y.
S7.
Compute shared secret key sk_{new}.
Authenticate gateway.
Compute authentication information S.
S9.
Decrypt the command and terminate the service.

Fig. 6. Service Termination Process

H. Service Accounting

Gateways may aggregate all the collected one-time signatures regularly and get the aggregated signatures according to the number recorded for different shared IoT providers. Specifically, for n different signatures $(\epsilon_{\mathrm{total,i}}, \rho_{\mathrm{total,i}})$ belong to the same IoT provider, where $i = 1, \ldots, n$, the new gateway can get aggregated signature by computing

$$(\epsilon_a, \rho_a) = \left(\sum_{i=1}^{n} \epsilon_{\mathrm{total,i}}(mod\ q), \sum_{i=1}^{n} \rho_{\mathrm{total,i}}(mod\ q) \right).$$

To prove the amount of signatures authenticated, gateways send $(pid_1, \theta_1, \ldots, pid_n, \theta_n, \epsilon_{\mathrm{a}}, \rho_{\mathrm{a}})$ to the central server. The central server first checks whether there exists the same pair (pid, θ), and then it verifies the aggregated signature as shown in **Algorithm** 3. If all the verifications pass, the central server pays bills to the corresponding IoT device provider and afterwards IoT device provider gives some economic incentive to gateways according to the mount of signatures they proved.

I. Entity Revocation

In our system, it is easy to achieve revocation. The central server can stop updating the secret keys of malicious gateways and shared IoT devices. So malicious shared IoT devices cannot conduct the mutual authentication with legitimate gateways and they cannot join the system any longer. The malicious gateways cannot be authenticated by legitimate users either to trick users out of signatures.

For users, if they use up their secret keys, their privilege to get IoT services will be naturally revoked. The central server can also revoke users' privilege beforehand by adding their pids into the bitmap and informing the gateways of this update.

V. Security Analysis

In this section, we will analyze the security features of our system in terms of message confidentiality, privacy preserving, signature unforgeability, and auditability. It shows that our scheme can effectively defend against potential attacks.

A. Message Confidentiality

Lemma 5.1: Any attacker cannot obtain any information from the encrypted messages.

Proof: Suppose the messages in service request $D = g_1^d, R' = g_1^{r'}$ can be obtained by an attacker, $\mathcal{A}$, and the attacker got the secret key through these messages. It means that the attacker can compute $g_1^{dr'}$, given D, R without knowing d, r', which contradicts with CDH assumption.

Then, suppose attacker $\mathcal{A}$ can get the communication messages in service discovery $y, profile, k_2, \beta, S, C$ by eavesdropping, there are two ways to compute the secret key sk. Because $sk = H_2(g_2^{H_2(\alpha||SK_j)H_2(\delta||SK_i)})$, $\mathcal{A}$ may compute sk through this equation directly. It's hard for $\mathcal{A}$ to know all the necessary information $\alpha, \delta, SK_j, SK_i$, so it is computationally infeasible to adversary $\mathcal{A}$ to obtain sk by this way. Another way is given y, k_2 to compute sk as follows:

$$y = (g_1^{H_2(i||TS_1)}\lambda)^{x_1} = g_1^{x_1 z}, y_1 = g_1^{(x_1 z)/z},$$
$$k = e(y_1, g_1) = g_2^{x_1}, sk = H_2(g_2^{x_1 x_2}),$$

where $g_1^z = g_1^{H_2(i||TS_1)}\lambda$. In other word, given $g_2^{x_1}, g_2^{x_2}$, $g_1^{x_1 z}, g_1^z$, the attacker need to compute $g_2^{x_1 x_2}, g_1^{x_1}$. This obviously contradicts to CDH and DCDH assumption. ■

B. Privacy Preserving

Lemma 5.2: Malicious gateways or external attackers cannot learn any information about users' privacy.

Proof: All the command related messages in both service request and command execution phases are encrypted. As proved in Lemma 4.1, attackers cannot know the service which is going to be executed by decrypting the messages. Besides, different OTSs are generated by using different secret keys and there is no relation between different secret keys, so the anonymity is ensured and attackers cannot infer who is requesting the services by the signature sent. ■

C. Signature Unforgeability

In our scheme, gateways should generate IBSs to prove that they are legitimate to the users, and the users also need to show OTSs to gateways to get services. So we will analyze signature unforgeability in terms of both IBS and OTS. First, if a malicious gateway can generate a valid IBS, it can pretend to be a legitimate gateway to cheat users of service credentials (i.e. OTSs). But the IBS in our system cannot be forged, which is proven in [44]. Then, if the one-time signature can be forged, malicious users can get service without paying money. But attackers are unable to generate a forged OTS either, and the corresponding proof is as follows:

Lemma 5.3: If the discrete logarithm problem is hard to solve on group G_1, any attacker cannot forge a valid one-time signature.

Proof: Suppose there exists an attacker, $\mathcal{A}$, who can break the unforgeability of one-time signature with non-negligible probability $Adv_{\mathcal{A}}$, we can find an algorithm $\mathcal{B}$ to solve the DLP.

Given (g, h) as input, $\mathcal{B}$ generates $(USK_1, UPK_1), \ldots, (USK_w, UPK_w)$ using (g, h) as shown in registration phase, and sends $UPK_1, \ldots, UPK_w$ and public parameters to $\mathcal{A}$. $\mathcal{A}$ makes one signature query for each key and $\mathcal{B}$ returns the valid signatures on random number $Q = \{\theta_1, \ldots, \theta_w\}$. Then, attacker $\mathcal{A}$ outputs a signature (ϵ, ρ) on a random number m, which is verified using UPK_i. Let $(\bar{\epsilon}, \bar{\rho})$ be the signature generated by $\mathcal{B}$ using USK_i. So we can have the following equation:

$$g_1^{\bar{\epsilon}_{\text{total}}} h^{\bar{\rho}_{\text{total}}} = g_1^{\epsilon_{\text{total}}} h^{\rho_{\text{total}}}.$$

Here we should consider two cases: (i) $m \notin Q$, (ii) $m \in Q$, but $(\epsilon, \rho) \neq (\bar{\epsilon}, \bar{\rho})$. Because the elements in $\bar{\rho}$ are the uniformly random values from $\{0, 1\}^{l_r}$, in case (i) the forged signature equals $(\bar{\epsilon}, \bar{\rho})$ with probability $1/C_{l_r}^{m}$ and the probability of case (ii) is $1 - 1/C_{l_r}^{m}$. Because the advantage of $\mathcal{A}$ forging a signature is $Adv_{\mathcal{A}}$, $\mathcal{A}$ can forge a signature satisfying case (ii) with probability $(1 - 1/C_{l_r}^{m})Adv_{\mathcal{A}}$. In such case, $\mathcal{B}$ obtain two different signatures, which allows $\mathcal{B}$ to solve DLP by computing $log_{g_1} h = (\bar{\rho}_{\text{total}} - \rho_{\text{total}})/(\epsilon_{\text{total}} - \bar{\epsilon}_{\text{total}})$ with a non-negligible advantage. It is contradicts to intractability of solving DLP. ■

D. Auditability

Audit mechanism is provided in our scheme to make it possible for central server to find the dishonest behaviors of malicious users and gateways in time.

Lemma 5.4: Malicious users dare not generate more than one signature using the same secret key.

Proof: As shown in **Algorithm** 1, every time a user generates a signature, $\lfloor m/2 \rfloor$ elements in SK will be exposed. Suppose a user generates two signatures using the same secret key but different random numbers θs, the best situation is exposing $\lfloor m/2 \rfloor + 1$ elements and the worst situation is exposing $2\lfloor m/2 \rfloor$ elements. Because the signatures are transported in plaintext, an attacker can generate extra $C_{\lfloor m/2 \rfloor + 1}^{\lfloor m/2 \rfloor} - 2 \sim C_{2\lfloor m/2 \rfloor}^{\lfloor m/2 \rfloor} - 2$ valid signatures to obtain IoT services. And central server can easily find these illegtimate signatures and make the malicious users pay for all these extra illegtimate usages. ■

Lemma 5.5: The gateways cannot increase the amount of signature they authenticate by forging and reusing.

Proof: The one-time signature is unforgeable, which is already proved in Lemma 4.3. Considering gateways reuse the collected signatures, the central server will find this kind of dishonest behaviors by checking whether there exists identical (pid, θ) pairs. ■

E. Comparison

We compare our scheme with several existing schemes in terms of security features. As shown in TABLE I, although all of these schemes can achieve secure access control, our scheme is the only one which can achieve access control, privacy preservation, offline server, mobility support and service accountability simultaneously. In particular, because PABE, SPSH, Heracles and DCapBAC are designed for IoT in traditional environment, mobility support and service accountability are not integrated into these systems. And due to the need of asking server for tokens, Heracles and DCapBAC require server to be online all the time. Besides, since tokens in Heracles and DCapBAC contain user identity and attributes in PABE are related to user, privacy can not be preserved in these schemes. Overall, our scheme has the best security features.

TABLE I
COMPARISON WITH OTHER ACCESS CONTROL SCHEMES

Scheme	Access Control	Privacy Preservation	Offline Server	Mobility Support	Service Accountability
PABE [20]	Yes	No	Yes	No	No
SPSH [21]	Yes	Yes	Yes	No	No
Heracles [23]	Yes	No	No	No	No
DCapBAC [25]	Yes	No	No	No	No
Our Scheme	Yes	Yes	Yes	Yes	Yes

VI. PERFORMANCE EVALUATION

In this section, we implement a prototype system and analyze its performance using our prototype. We use Google Nexus 5 (2.3GHz CPU, 2G RAM) as the subject devices own by users, Google Cloud with intel Xeon 2.5GHz CPU as the central server, and simulate gateways and shared IoT devices by Banana Pi R1 (1.2GHz CPU, 1G RAM). In our evaluation, subject devices/shared IoT devices connect with gateways through WiFi and gateways communicate with the central server through Ethernet. To accomplish our evaluation, we use Pairing-Based Cryptography (PBC) library and OpenSSL library in Google Cloud and Banana Pi and Java Pairing-Based Cryptography (JPBC) library and AndroidOpenSSL library in Google Nexus 5.

A. Key Management Overhead

We first test the OTS related key management overhead. In our scheme, the central server must generate massive OTS related secret keys and users also need to use hash function to generate all of their secret keys. As shown in Fig. 7, the key generation only costs users several milliseconds and it costs the central server much more time due to the exponent arithmetic on G_1. But this operation can be executed in parallel, so the cost is acceptable for the central server.

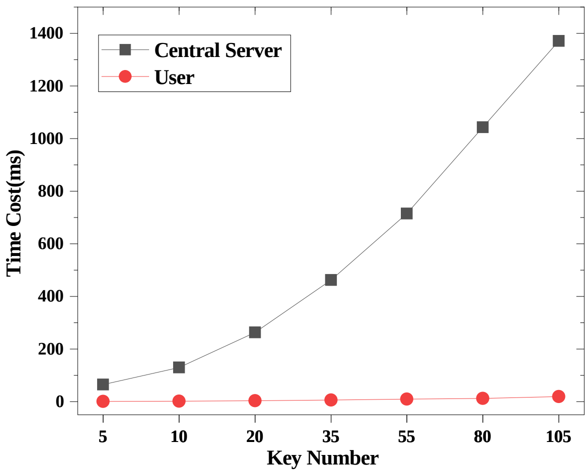


Fig. 7. Key management overhead

B. Discovery Overhead

In our system, gateways and shared IoT devices are not fully-trusted, so the mutual authentication is processed in discovery phase. Fig. 8 (a) presents the computation overhead of gateways and IoT devices. We can see that we put heavy operations (e.g., pairing) in gateways and it can finish computation in 64ms. The computations in IoT devices are the lighter operations (e.g., exponent and hash), and thus the time cost in IoT devices is only 18% of that in gateways. So it is feasible to apply our scheme to the shared IoT devices with weaker CPUs. We also test the *authentication latency* (total computation overhead plus communication overhead) in real-world scenario using our prototype. The result shows that the discovery phase can be accomplished in 91ms, which is acceptable considering this phase is not often execusted and it has no relation with user experience.

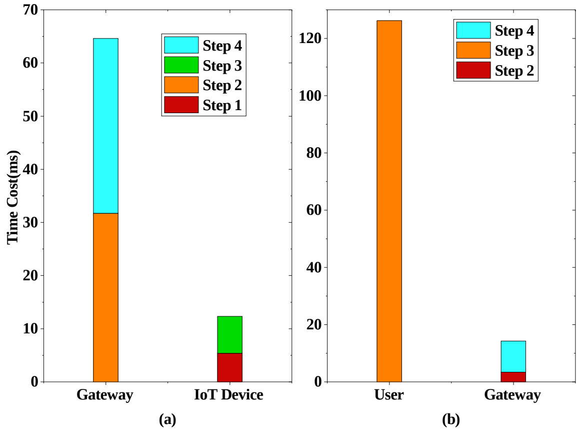


Fig. 8. Discovery and request overhead

C. Request and Execution Overhead

1) Overhead Analysis: Next, we test the performance of request, execution and termination, which are directly related to user experience. In request phase, subject devices are required to conduct OTS generation, IBS verification and Enc process each one time, and Object devices need to conduct OTS verification, IBS generation and Dec for one time correspondingly. TABLE II shows the computation cost for these processes. We set security parameters in OTS as $m = 19, l_r = 32, n = 16$. Because OTS generation only needs m times lookup operation, subject devices can generate a OTS in 0.8ms, which is very fast. It is also efficient for object devices to verify OTSs and generate IBSs. Because JPBC library is not very efficient for computation on group G_1, so the time cost for subject devices to verify IBS is a little high.

TABLE II
COMPUTATION COST FOR CRYPTOGRAPHIC PROCESSES

Device	Processes	Operations	Time (ms)
Subject Devices	OTS Generation	m lookup	0.8
	IBS Verification	2 M+2 E+2 H+1 DH	84.9
	DH Key Exchange	1 E	40.5
	Enc	AES-256	0.06/KB
Object Devices	OTS Verification	m lookup+m comparison +18 A+2 E+1 M	7.6
	IBS Generation	1 A+1 M +1 E	3.4
	DH Key Exchange	1 E	3.3
	Dec	AES-256	0.036/KB

* A, M and E represent the addition, multiplication and exponentiation operation on group G_1. We denote H, DH as hash and DH key exchange operation respectively.

Fig. 8 (b) shows the the detailed time cost of users and gateways. In step 1, users only generate a random number, whose time cost is negligible, so it is not shown in the figure. However, step 3 costs about 126 ms with proportions of each operations as 0.6% in OTS generation, 67.3% in IBS verification, and 32.1% in DH key exchange. The computation overhead in gateway is lower and the proportions of its time cost are: 53.1% in OTS verification, 23.8% in IBS generation, and 23.1% in DH key exchange. The symmetric encryption is very fast in both subject devices and object devices with speed 0.036 ms/KB and 0.06 ms/KB, respectively, which is negligible to other operations. The time cost ratio of IBS and DH key exchange operations is very large, so we can find better IBS and secret key negotiation algorithm to improve our scheme in the future.

In the execution phase, gateways and shared IoT devices only conduct symmetric encryption which is very fast and can be overlooked. Similarly, we utilize our prototype system to measure the *execution latency* (the time cost from users sending first message in request phase to shared IoT devices executing the commands). The execution latency is about 159 ms, which means that users can hardly sense the execution latency.

2) Comparison: In order to show the advantages of our scheme, we also compare our scheme with PABE, SPSH, Heracles and DCapBAC in terms of operations and overhead in request and execution phases. Note that the number of punctured attributes and the security parameter d in the PABE are set as 0 and 5 respectively. We consider that there is only one set of 5 attributes can satisfy the corresponding access policy in SPSH. We use an elliptic curve with 160-

TABLE III
REQUEST AND EXECUTION OPERATION COMPARISON

Scheme	Request			Execution		
	User	Gateway	Server	User	Gateway	IoT Device
PABE [20]	-	-	-	Pt-CP-ABE	-	-
SPSH [21]	-	-	-	PH-CP-ABE	-	-
Heracles [23]	RSA,$R\bar{S}A$	-	RSA,$R\bar{S}A$	RSA,$R\bar{S}A$	-	RSA,$R\bar{S}A$
DCapBAC [25]	ECDSA	-	ECDSA	-	-	$EC\bar{D}SA^2$
Our Scheme	$I\bar{B}S$,OTS,Enc	IBS,Dec,$O\bar{T}S$	-	-	Enc,Dec	Dec,Enc

* RSA/$R\bar{S}A$, ECDSA/$EC\bar{D}SA$, IBS/$I\bar{B}S$ and OTS/$O\bar{T}S$ represents corresponding algorithm signing/verifying process respectively. $EC\bar{D}SA^2$ means corresponding algorithm are conducted 2 times. Pt-CP-ABE and PH-CP-ABE are improved CP-ABE algorithms used in literatures [20] and [21], respectively.

bit group order to implement our scheme. And we implement 1024-bit RSA in Heracles and 192-bit ECDSA in DCapBAC, which have the approximate security level of our scheme. Besides, instead of utilizing a security model where gateways cannot be fully trusted, these compared schemes either have no gateways or have fully trusted gateways. Therefore, for fairness consideration, we also compare these schemes with *Our Scheme-WHG*, where the gateways are fully trusted and thus IBS related operations are exempted.

TABLE III and TABLE IV show the operations needed and overhead cost in the different schemes respectively. We can see that although PABE and SPSH only need user to conduct one CP-ABE operation, CP-ABE is too heavy for users' subject devices and it costs more than 1300 ms to finish this operation. So PABE and SPSH have the most execution latency in all six schemes. In Heracles, users are required to execute twice RSA verification and generation, and server and IoT device are only required to execute once. The time cost in user side is 210 ms, which takes up 73.6% of the execution latency and is much larger than our scheme. In DCapBAC, IoT device needs to verify the ECDSA signatures generated by the user and server. DCapBAC performs better than other these schemes, and also has lower execution latency than our scheme. But with the same security model, the execution latency of Our Scheme-WHG is only 18.4ms, which is more than 5 times faster than DCapBAC. Therefore, we can draw the conclusion our scheme has the highest efficiency among these schemes in a sharing economy environment.

TABLE IV
REQUEST AND EXECUTION OVERHEAD COMPARISON

Scheme	Computation Overhead				Execution Latency
	User	Server	Gateway	IoT Device	
PABE [20]	1387.5	-	-	-	1439.5
SPSH [21]	1339.0	-	-	-	1391.2
Heracles [23]	210.0	0.6	-	12.7	285.3
DCapBAC [25]	3.0	0.9	-	37.3	103.2
Our Scheme-WHG	0.8	-	7.6	< 0.1	18.4
Our Scheme	125.4	-	13.7	< 0.1	159.1

D. Termination Overhead

The computation overhead in termination phase is related to that in discovery and request phase, so we do not analyze it here. Fig. 6 gives the *termination latency* (the time cost from users sending first message in termination phase to users receiving success feedback message) of type A devices. Note that users need to send num OTSs to gateways and termination latency increases from 155 ms to 307 ms with num growing from 2 to 20 when gateways verify OTSs individually. So when the signature number is too big, our system will lose efficiency, which is the situation that we do not expect. So the OTS in our scheme can also support batch verification using the small exponents tests proposed in [45] and we conduct the batch verification in our system where the length of random exponents is set as 30. As shown in Fig. 9, the terminal latency can be decreased by 4.8%, 13.1%, 25.5%, 38.1%, and 46.6% when the num is set as 2, 4, 8, 14, and 20, respectively. We can see that the batch verification can largely improve efficiency of our system when the num is very big. For type B devices, because they are terminated automatically when they finish the services, the termination latency is 0.

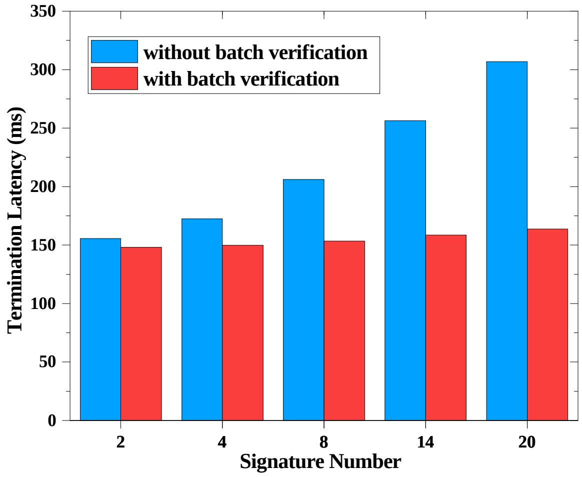


Fig. 9. Termination Latency

E. Accounting Overhead

To make it easy for the central server for accounting, gateways aggregate the OTSs collected regularly and the central server only needs to verify the aggregated signatures. Here we measure the accounting overhead from signature aggregation and aggregated signature verification. As shown in

Fig. 10, when the number of signatures increases from 10K to 100K, the time cost of signature aggregation and aggregated signature verification varies from 55.3 ms to 612.3 ms and 278.7 ms to 2860.5 ms, respectively. It indicates that the accounting phase is very fast and it only brings a little to both gateways and central server.

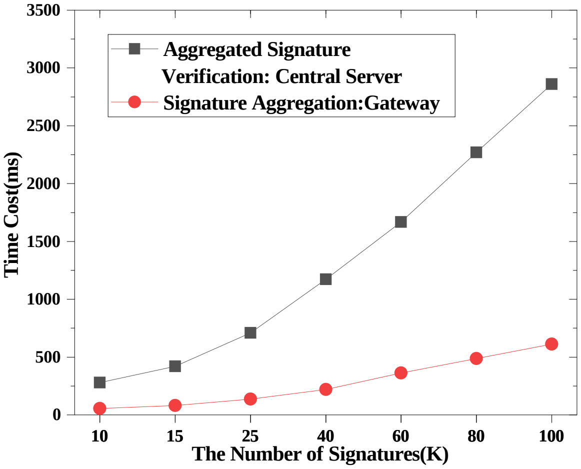


Fig. 10. Accounting Overhead

VII. Conclusion

In this paper, we designed an efficient and secure access control scheme for IoT in the sharing economy environment. Our design effectively supports service accountability, privacy preservation, and information feedback. By adopting one-time signatures, anonymous authentication can be achieved and one-time signatures are considered as trusted credentials used in service accounting. The computation overhead in service accounting can be largely reduced by making gateways aggregate collected signatures. Our proposed protocols are able to deal with the mobility problem of shared IoT devices and let IoT providers collect some feedback information without disclosing users' privacy. Our security analysis shows that our scheme can successfully defend against potential attacks, and the results of experiments conducted in the implemented prototype system demonstrate that our scheme also ensures good efficiency.

Acknowledgment

The authors sincerely thank the anonymous referees for their invaluable suggestions that have led to the present improved version from the original manuscript. This work is supported in part by National Natural Science Foundation of China under Grant No. 61972371, and Youth Innovation Promotion Association Chinese Academy of Sciences (CAS) under Grant No. 2016394.

References

[1] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet of Things journal*, vol. 1, no. 1, pp. 22–32, 2014.

[2] J. Dizdarević, F. Carpio, A. Jukan, and X. Masip-Bruin, "A survey of communication protocols for internet of things and related challenges of fog and cloud computing integration," *ACM Computing Surveys (CSUR)*, vol. 51, no. 6, p. 116, 2019.

[3] I. Yaqoob, I. A. T. Hashem, A. Ahmed, S. A. Kazmi, and C. S. Hong, "Internet of things forensics: Recent advances, taxonomy, requirements, and open challenges," *Future Generation Computer Systems*, vol. 92, pp. 265–275, 2019.

[4] Y.-W. Kuo, C.-L. Li, J.-H. Jhang, and S. Lin, "Design of a wireless sensor network-based IoT platform for wide area and heterogeneous applications," *IEEE Sensors Journal*, vol. 18, no. 12, pp. 5187–5197, 2018.

[5] M. T. Lazarescu, "Design of a WSN platform for long-term environmental monitoring for IoT applications," *IEEE Journal on emerging and selected topics in circuits and systems*, vol. 3, no. 1, pp. 45–54, 2013.

[6] K. Ren, Q. Wang, C. Wang, Z. Qin, and X. Lin, "The security of autonomous driving: Threats, defences, and future directions," *Proceedings of the IEEE*, vol. 108, no. 2, pp. 357–372, 2020.

[7] X. Shen, R. Fantacci, and S. Chen, "Internet of vehicles," *Proceedings of the IEEE*, vol. 108, no. 2, pp. 242–245, 2020.

[8] S. Li, K. Xue, Q. Yang, and P. Hong, "PPMA: Privacy-preserving multi-subset data aggregation in smart grid," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 2, pp. 462–471, 2018.

[9] K. Xue, B. Zhu, Q. Yang, D. S. Wei, and M. Guizani, "An efficient and robust data aggregation scheme without a trusted authority for smart grid," *IEEE Internet of Things Journal*, 2019, Online, DOI: 10.1109/JIOT.2019.2961966.

[10] S. Li, K. Xue, D. S. Wei, H. Yue, N. Yu, and P. Hong, "SecGrid: A secure and efficient SGX-enabled smart grid system with rich functionalities," *IEEE Transactions on Information Forensics and Security*, vol. 15, no. 1, pp. 1318–1330, 2020.

[11] S. Li, X. Zhang, K. Xue, L. Zhou, and H. Yue, "Privacy-preserving prepayment based power requestand trading in smart grid," *China Communications*, vol. 15, no. 4, pp. 14–27, 2018.

[12] "The sharing economy: Understanding the opportunities for growth," https://newsroom.mastercard.com/eu/files/2017/06/Mastercard_Sharing-Economy_v7.compressed2.pdf, 2017.

[13] "Bird cruiser," https://www.bird.co/.

[14] A. B. Sherif, K. Rabieh, M. M. Mahmoud, and X. Liang, "Privacy-preserving ride sharing scheme for autonomous vehicles in big data era," *IEEE Internet of Things Journal*, vol. 4, no. 2, pp. 611–618, 2017.

[15] Y. Benazzouz, C. Munilla, O. Gunalp, M. Gallissot, and L. Gurgen, "Sharing user IoT devices in the cloud," in *Proceedings of 2014 IEEE World Forum on Internet of Things (WF-IoT)*. IEEE, 2014, pp. 373–374.

[16] S. Cherrier, Z. Movahedi, and Y. M. Ghamri-Doudane, "Multi-tenancy in decentralised IoT," in *Proceedings of 2015 IEEE World Forum on Internet of Things (WF-IoT)*. IEEE, 2015, pp. 256–261.

[17] D. He, J. Bu, S. Zhu, S. Chan, and C. Chen, "Distributed access control with privacy support in wireless sensor networks," *IEEE Transactions on Wireless Communications*, vol. 10, no. 10, pp. 3472–3481, 2011.

[18] B. Panja, S. K. Madria, and B. Bhargava, "A role-based access in a hierarchical sensor network architecture to provide multilevel security," *Computer Communications*, vol. 31, no. 4, pp. 793–806, 2008.

[19] S. Misra and A. Vaish, "Reputation-based role assignment for role-based access control in wireless sensor networks," *Computer Communications*, vol. 34, no. 3, pp. 281–294, 2011.

[20] T. V. X. Phuong, R. Ning, C. Xin, and H. Wu, "Puncturable attribute-based encryption for secure data delivery in internet of things," in *Proceedings of 2018 IEEE International Conference on Computer Communications (INFOCOM)*. IEEE, 2018, pp. 1511–1519.

[21] Y. Zhang, D. Zheng, and R. H. Deng, "Security and privacy in smart health: efficient policy-hiding attribute-based access control," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 2130–2145, 2018.

[22] X. Wang, J. Zhang, E. M. Schooler, and M. Ion, "Performance evaluation of attribute-based encryption: Toward data privacy in the IoT," in *Proceedings of 2014 IEEE International Conference on Communications (ICC)*. IEEE, 2014, pp. 725–730.

[23] Q. Zhou, M. Elbadry, F. Ye, and Y. Yang, "Heracles: Scalable, fine-grained access control for internet-of-things in enterprise environments," in *Proceedings of 2018 IEEE International Conference on Computer Communications (INFOCOM)*. IEEE, 2018, pp. 1772–1780.

[24] P. N. Mahalle, B. Anggorojati, N. R. Prasad, and R. Prasad, "Identity authentication and capability based access control (IACAC) for the internet of things," *Journal of Cyber Security and Mobility*, vol. 1, no. 4, pp. 309–348, 2013.

[25] J. L. Hernández-Ramos, A. J. Jara, L. Marín, and A. F. Skarmeta Gómez, "DCapBAC: embedding authorization logic into smart things through ECC optimizations," *International Journal of Computer Mathematics*, vol. 93, no. 2, pp. 345–366, 2016.

[26] L. Xiao, X. Wan, X. Lu, Y. Zhang, and D. Wu, "Iot security techniques based on machine learning: How do iot devices use ai to enhance security?" *IEEE Signal Processing Magazine*, vol. 35, no. 5, pp. 41–49, 2018.

[27] P. Kumar, A. Braeken, A. Gurtov, J. Iinatti, and P. H. Ha, "Anonymous secure framework in connected smart home environments," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 4, pp. 968–979, 2017.

[28] N. Z. Aitzhan and D. Svetinovic, "Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 5, pp. 840–852, 2018.

[29] T. Song, R. Li, B. Mei, J. Yu, X. Xing, and X. Cheng, "A privacy preserving communication protocol for IoT applications in smart homes," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1844–1852, 2017.

[30] G. M. Zaverucha and D. R. Stinson, "Short one-time signatures," *Advances in Mathematics of Communications*, vol. 5, no. 3, pp. 473–488, 2011.

[31] W. Li, K. Xue, Y. Xue, and J. Hong, "TMACS: A robust and verifiable threshold multi-authority access control system in public cloud storage," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 5, pp. 1484–1496, 2016.

[32] K. Xue, J. Hong, Y. Ma, D. S. Wei, P. H. Hong, and N. Yu, "Fog-aided verifiable privacy preserving access control for latency-sensitive data sharing in vehicular cloud computing," *IEEE Network*, vol. 32, no. 3, pp. 7–13, 2018.

[33] J. Hong, K. Xue, Y. Xue, W. Chen, D. S. Wei, N. Yu, and P. Hong, "TAFC: Time and attribute factors combined access control for time-sensitive data in public cloud," *IEEE Transactions on Services Computing*, vol. 13, no. 1, pp. 158–171, 2020.

[34] Y. Xue, K. Xue, N. Gai, J. Hong, D. S. Wei, and P. Hong, "An attribute-based controlled collaborative access control scheme for public cloud storage," *IEEE Transactions on Services Computing*, vol. 14, no. 11, pp. 2927–2942, 2019.

[35] W. Zhou, Y. Jia, Y. Yao, L. Zhu, L. Guan, Y. Mao, P. Liu, and Y. Zhang, "Discovering and understanding the security hazards in the interactions between iot devices, mobile apps, and clouds on smart home platforms," in *28th USENIX Security Symposium (USENIX Security)*, 2019, pp. 1133–1150.

[36] E. Luo, M. Z. A. Bhuiyan, G. Wang, M. A. Rahman, J. Wu, and M. Atiquzzaman, "PrivacyProtector: Privacy-protected patient data collection in IoT-based healthcare systems," *IEEE Communications Magazine*, vol. 56, no. 2, pp. 163–168, 2018.

[37] Q. Wang, Y. Zhang, X. Lu, Z. Wang, Z. Qin, and K. Ren, "Real-time and spatio-temporal crowd-sourced social network data publishing with differential privacy," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 4, pp. 591–606, 2018.

[38] J. Liu, C. Zhang, and Y. Fang, "EPIC: A differential privacy framework to defend smart homes against internet traffic analysis," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1206–1217, 2018.

[39] S. Hu, L. Y. Zhang, Q. Wang, Z. Qin, and C. Wang, "Towards private and scalable cross-media retrieval," *IEEE Transactions on Dependable and Secure Computing*, 2019, Online, DOI: 10.1109/TDSC.2019.2926968.

[40] V. Costan and S. Devadas, "Intel SGX explained," *IACR Cryptology ePrint Archive*, vol. 2016, no. 086, pp. 1–118, 2016.

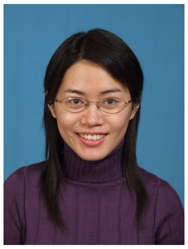
[41] P. Kumar, A. Gurtov, J. Iinatti, M. Ylianttila, and M. Sain, "Lightweight and secure session-key establishment scheme in smart home environments," *IEEE Sensors Journal*, vol. 16, no. 1, pp. 254–264, 2016.

[42] F. Bao, R. H. Deng, and H. Zhu, "Variations of diffie-hellman problem," in *International Conference on Information and Communications Security (ICICS)*. Springer, 2003, pp. 301–312.

[43] P. Barreto, B. Libert, N. McCullagh, and J.-J. Quisquater, "Efficient and provably-secure identity-based signatures and signcryption from bilinear maps," *Advances in Cryptology-ASIACRYPT 2005*, pp. 515–532, 2005.

[44] M. Bellare, C. Namprempre, and G. Neven, "Security proofs for identity-based identification and signature schemes," *Journal of Cryptology*, vol. 22, no. 1, pp. 1–61, 2009.

[45] M. Bellare, J. A. Garay, and T. Rabin, "Fast batch verification for modular exponentiation and digital signatures," *Advances in Cryptology-EUROCRYPT '98*, vol. 1403, pp. 236–250, 1998.



Yu Liu received her B.S. degree in the Department of Management, Anhui University, in 2003, and her M.S. degree in the School of Management, Hefei University of Technology, in 2006. She is currently an associate professor in School of Economics and Management, Hefei University. Her research interests include modern logistics technology, E-commerce security and network security.

Kaiping Xue (M'09-SM'15) received his bachelor's degree from the Department of Information Security, University of Science and Technology of China (USTC), in 2003 and received his Ph.D. degree from the Department of Electronic Engineering and Information Science (EEIS), USTC, in 2007. From May 2012 to May 2013, he was a postdoctoral researcher with the Department of Electrical and Computer Engineering, University of Florida. Currently, he is an Associate Professor in the School of Cyber Security and the Department of EEIS, USTC. His research interests include next-generation Internet, distributed networks and network security. Dr. Xue has authored and co-authored more than 80 technical papers in the areas of communication networks and network security. His work won best paper awards in IEEE MSN 2017, IEEE HotICN 2019, and best paper runner-up award in IEEE MASS 2018. He serves on the Editorial Board of several journals, including the IEEE Transactions on Wireless Communications (TWC), the IEEE Transactions on Network and Service Management (TNSM), Ad Hoc Networks, IEEE Access and China Communications. He has also served as a guest editor of IEEE Journal on Selected Areas in Communications (JSAC) and a lead guest editor of IEEE Communications Magazine. He is serving as the Program Co-Chair for IEEE IWCMC 2020 and SIGSAC@TURC 2020. He is an IET Fellow and an IEEE Senior Member.

Peixuan He received the B.S. degree from the department of Information Security, University of Science and Technology of China (USTC), in 2017. He is currently a graduated student in Information Security from the Department of Electronic Engineering and Information Science (EEIS), USTC. His research interests include network security protocol design and analysis.

David S.L. Wei (SM'07) received his Ph.D. degree in Computer and Information Science from the University of Pennsylvania in 1991. He is currently a Full Professor of Computer and Information Science Department at Fordham University. From May 1993 to August 1997 he was on the Faculty of Computer Science and Engineering at the University of Aizu, Japan (as an Associate Professor and then a Full Professor). Dr. Wei has authored and co-authored more than 120 technical papers in the areas of distributed and parallel processing, wireless networks and mobile computing, optical networks, peer-to-peer communications, cognitive radio networks, big data, cloud computing, and IoT in various archival journals and conference proceedings. He served on the program committee and was a session chair for several reputed international conferences. He was a lead guest editor of IEEE Journal on Selected Areas in Communications for the special issue on Mobile Computing and Networking, a lead guest editor of IEEE Journal on Selected Areas in Communications for the special issue on Networking Challenges in Cloud Computing Systems and Applications, a guest editor of IEEE Journal on Selected Areas in Communications for the special issue on Peer-to-Peer Communications and Applications, a lead guest editor of IEEE Transactions on Cloud Computing for the special issue on Cloud Security, a guest editor of IEEE Transactions on Big Data for the special issue on Trustworthiness in Big Data and Cloud Computing Systems, and a lead guest editor of IEEE Transactions on Big Data for the special issue on Edge Analytics in the Internet of Things. He also served as an Associate Editor of IEEE Transactions on Cloud Computing, 2014-2018, and an Associate Editor of Journal of Circuits, Systems and Computers, 2013-2018. He is presently an editor of IEEE Journal on Selected Areas in Communications for the Series on Network Softwarization & Enablers and a lead guest editor of IEEE Journal on Selected Areas in Communications for the special issue on Leveraging Machine Learning in SDN/NFV-based Networks. Currently, Dr. Wei focuses his research efforts on cloud and edge computing, IoT, big data, machine learning, and cognitive radio networks.

Mohsen Guizani (S'85-M'89-SM'99-F'09) received the B.S. (with distinction) and M.S. degrees in electrical engineering, the M.S. and Ph.D. degrees in computer engineering from Syracuse University, Syracuse, NY, USA, in 1984, 1986, 1987, and 1990, respectively. He is currently a Professor at the CSE Department in Qatar University, Qatar. Previously, he served in different academic and administrative positions at the University of Idaho, Western Michigan University, University of West Florida, University of Missouri-Kansas City, University of Colorado-Boulder, and Syracuse University. His research interests include wireless communications and mobile computing, computer networks, mobile cloud computing, security, and smart grid. He is currently the Editor-in-Chief of the IEEE Network Magazine, serves on the editorial boards of several international technical journals and the Founder and Editor-in-Chief of Wireless Communications and Mobile Computing journal (Wiley). He is the author of nine books and more than 500 publications in refereed journals and conferences. He guest edited a number of special issues in IEEE journals and magazines. He also served as a member, Chair, and General Chair of a number of international conferences. Throughout his career, he received three teaching awards and four research awards. He also received the 2017 IEEE Communications Society WTC Recognition Award as well as the 2018 Ad Hoc Technical Committee Recognition Award for his contribution to outstanding research in wireless communications and Ad-Hoc Sensor networks. He was the Chair of the IEEE Communications Society Wireless Technical Committee and the Chair of the TAOS Technical Committee. He served as the IEEE Computer Society Distinguished Speaker and is currently the IEEE ComSoc Distinguished Lecturer. He is a Fellow of IEEE and a Senior Member of ACM.