Achieving Privacy Preservation and Billing via Delayed Information Release

Chunqiang Hu, *Member, IEEE, ACM*, Xiuzhen Cheng, *Fellow, IEEE, Member, ACM*, Zhi Tian, *Fellow, IEEE*, Jiguo Yu, *Senior Member, IEEE*, and Weifeng Lv

***Abstract*—Many applications such as smart metering and location based services pose strong privacy requirements but achieving privacy protection at the client side is a non-trial problem as payment for the services must be computed by the server at the end of each billing period. In this paper, we propose a privacy preservation and billing scheme termed PPDIR based on delayed information release. PPDIR relies on a novel group signature mechanism and the asymmetric Rabin cryptosystem to protect the privacy of the clients and their requests, to achieve accountability and non-repudiation, and to shift the computational complexity to the server side. It adopts a secret token for anonymity and the token is updated for each client at the beginning of each billing period and securely released only to the server at the end of the billing period. Such a strategy can prevent the server from linking a client's requests made at different billing periods. It also prevents any adversary from linking any request to any client. Note that the server is able to figure out all requests made by a client within a billing period after receiving the delayed token, which is unavoidable for billing purpose. We prove the security properties of the group signature scheme, and analyze the security strength of PPDIR. Our study indicates that PPDIR can achieve privacy-preservation, confidentiality, non-repudiation, accountability, and other security objectives. We also evaluate the performance of our scheme in terms of communication and computational overheads.**

***Index Terms*—Privacy preservation, group signature, billing, delayed information release, smart grids, location based services.**

Manuscript received April 9, 2019; revised July 31, 2020; accepted February 25, 2021; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor Y. Chen. Date of publication March 12, 2021; date of current version June 16, 2021. This work was supported in part by the National Key Research and Development Project under Grant 2019YFB2102600; in part by the National Natural Science Foundation of China under Grant 62072065, Grant 61932006, Grant 61771289, and Grant 61832012; in part by the Key Project of Technology Innovation and Application Development of Chongqing under Grant cstc2019jscx-mbdxX0044; in part by the Natural Science Foundation of Chongqing, China, under Grant cstc2018jcyjA3041; in part by the Overseas Returnees Innovation and Entrepreneurship Support Program of Chongqing under Grant cx2018015 and Grant cx2020004; and in part by the National Science Foundation of the U.S. under Grant CNS-1704397, Grant CNS-1704274, Grant ECCS-1407986, Grant IIS-1741279, and Grant IIS-1741338. Major parts of this work were completed when the first author was a Ph.D. student at The George Washington University. *(Corresponding authors: Chunqiang Hu; Jiguo Yu.)*

Chunqiang Hu is with the School of Big Data & Software Engineering, Chongqing University, Chongqing 400044, China, and also with the Key Laboratory of Dependable Service Computing in Cyber Physical Society, Ministry of Education, Chongqing University, Chongqing 400044, China (e-mail: chu@cqu.edu.cn).

Xiuzhen Cheng is with the Department of Computer Science, The George Washington University, Washington, DC 20052 USA (e-mail: cheng@gwu.edu).

Zhi Tian is with the Electrical and Computer Engineering Department, George Mason University, Fairfax, VA 22030 USA (e-mail: ztian1@gmu.edu).

Jiguo Yu is with the School of Computer Science and Technology, Qilu University of Technology (Shandong Academy of Sciences), Jinan 250353, China, and also with the Shandong Laboratory of Computer Networks, Jinan 250014, China (e-mail: jiguoyu@sina.com).

Weifeng Lv is with the State Key Laboratory of Software Development Environment, Beihang University, Beijing 100191, China, and also with the Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University, Beijing 100191, China (e-mail: lwf@nlsde.buaa.edu.cn).

Digital Object Identifier 10.1109/TNET.2021.3063102

I. INTRODUCTION

MANY real world applications can be modeled with a simple client-server architecture, in which clients send requests to a server, get services from the server on demand, and pay the server for the received services. For example, as shown in Fig. 1, location based services (LBS) allow mobile users to contact LBS servers via their smart devices to get context information such as the closest gas station or the most highly rated Asia restaurant within 5 miles. Smart metering (Fig. 2) can also be regarded as following this simple client-server model as smart meters (clients) periodically report the fine-grained electricity usage data to utility companies (servers), though a utility company does not have to respond to each report (a null response from the server). Such applications share the following properties:

- The data (a LBS request or a smart meter reading) sent from a client to the server can reveal the private information of the client. For examples, LBS requests could be exploited to trace a mobile user and infer the user's private information such as race and age as the request contents contain the location and preferences of the user [1]–[5]; smart meter readings can be explored to figure out the presence/absence and habit of a residence as the fine-grained utility data closely reflect the activities of the residence [6], [7]. Such a privacy disclosure triggers many security and safety concerns, and even physical attacks to the clients.
- Protecting the privacy of the clients and the confidentiality of the requests can be achieved via traditional and advanced cryptographic primitives but unfortunately existing research exhibits the following disadvantages of such approaches: (i) the computational overhead is prohibitively high as in general advanced crypto tools such as homomorphic encryption is employed but the client devices in our application scenarios are typically relatively "dumb" (smart phones, smart meters, etc.) [5], [8]–[11]; (ii) completely hiding the clients via anonymous communications to prevent the server and adversaries from linking the requests made by the same client renders the traditional billing, which computes bills by the server based on the amount of offered services, impossible, and billing by clients with price functions publicized by the server causes further security problems [12]–[14];

Fig. 1. A location based service system architecture.

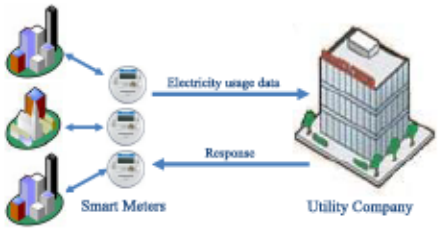


Fig. 2. The smart metering architecture for fine-grained electricity usage data collection.

(iii) many proposed approaches such as privacy-preserving data aggregation for smart metering [15]–[17] actually could not achieve the desired objectives as the coarse-grained aggregated data gets rid of the advantages provided by fine-grained utility data for smart grid services, which implies a sacrifice of service with privacy protection.

These observations motivate us to study the following novel problem under a simple client-server model: *how to protect the privacy of the clients (identity and data) while simultaneously getting services from the server and paying the server for the received services in a billing period?* This is a non-trivial problem as our design has to consider the following security challenges and objectives: (i) the (identity and data) privacy of the clients needs to be protected while the legitimacy of each request should be verifiable; (ii) the confidentiality of each request should be protected; (iii) the monthly bill for each client should be computable at the server side though the privacy of the client needs to be protected; (iv) the transactions from different billing periods must be completely unlinkable by any entity; (v) the transactions from the same billing period can be linked together only by the server for billing purpose when a bill needs to be generated; (vi) the computational and communication overhead must be low at the client side; and (vii) the solution approach much achieves nonrepudiation and accountability as a client should never be able to deny the requests/services it has made/received and the server should never be able to forge a service not requested by any client without being detected.

To tackle these challenges, we propose a novel accountable scheme to provide **P**rivacy **P**reservation and billing via **D**elayed **I**nformation **R**elease (PPDIR) for client-server based applications. PPDIR employs (i) a novel lightly-weighted group signature scheme such that the signature of each client request is verifiable (the request is from a legitimate client) but the server is not able to figure out who the client is without further information; (ii) the computationally asymmetric Rabin's public-key cryptosystem to encrypt a client request with the server's public key to provide request confidentiality; (iii) AES to protect the server responses; and (iv) a one-way hash chain seeded by a token that changes at the beginning of each billing period such that the server can link the requests made by the same client in the same billing period only when the token is securely released to the server at the end of the current billing period while no one else can link any request from any billing period at any time. We carry out a rigorous analysis to investigate the security properties of the proposed group signature mechanism, and the security strength and computational/communication overhead of PPDIR.

An important consideration in our design is the low complexity at the client side. Our novel group signature scheme ensures that one signing key pair can be used to sign as many requests as the client wishes while traditional ones [18], [19] require one signing key pair per message; thus our mechanism incurs significantly low storage and key management overheads at the client. Moreover, Rabin's public-key cryptosystem combined with AES help shift the computational complexity from the client side to the server, as Rabin encryption (at client side) is much simpler than its decryption operation (at server side), i.e., a client only needs to perform one modular multiplication plus AES to protect confidentiality of its request and response.

The rest of this paper is organized as follows: Section II presents the related work. Section III outlines our system and security models as well as the preliminary knowledge employed by PPDIR. Section IV and V respectively detail PPDIR and its security properties and strengths. The performance of PPDIR in terms of computational and communication overheads are reported in Section VI. The discussions on PPDIR and our future research are presented in Section VII. Conclusions are presented in Section VIII.

II. Related Works

The survey by Gigya [20] revealed that data privacy was concerned by 90% of people. In real-world, it is a common practice for human beings to submit data and pay for the requested services. Nevertheless, existing research considered how to protect either data and user identity privacy [21]–[23], or the real-time pricing functions [24]–[28], overlooking how users are billed, though some claimed that bills can be computed by users based on the published pricing function; Nevertheless, such an approach introduces more security concerns and definitely is undesirable by the service providers. In this paper, we propose a mechanism that can protect data/identity privacy while guaranteeing that bills can be computed by the service providers based on the fine-grained data submitted by users. To our best knowledge, this is the first work simultaneously achieving the above goals.

Data and identity privacy protection have been extensively studied without considering billing in diverse contexts for various applications. Identity protection is usually realized via anonymization [29] but data protection is more sophisticated. Perturbation based techniques such as k-anonymity [30], l-diversity [31], t-closeness [32], [33], and differential privacy [34], have been tailored for privacy protection in smart grids [35], [36], social networks [5], [37], [38], location-based services [39], [40], privacy-perserving localization [41], [42], e-healthcare [43]–[45], Internet of Things [46], [47], just to name a few. Such approaches usually rely on added noise to protect data, resulting in decreased quality-of-services and user experiences as well as inaccurate bills. Cryptography-based techniques [48], [49] such as homomorphic encryption [50], secret sharing [51], and functional encryption [52] can provide privacy protection but the involved computational burden is

Fig. 3. A communication architecture.

prohibitively high, rendering them impractical at the client side to encrypt the data and at the server side to compute the bills in practice. Another popular approach for data privacy protection is data aggregatioin [53]–[56], which submits only coarse-grained or summary data to the servers, causing fine-grained information loss and accordingly decreased qualify of services and failed bill computation based on real-time pricing at the server side. Our work can simultaneously achieve privacy protection as well as traditional billing based on fine-grained data.

III. Models and Preliminaries

A. System Model

As shown in Fig. 3, our system model consists of four major entities: Key Generation Center (KGC), Client, Data center, and Server. In the following we briefly summarize the major functions of each entity.

- *The Key Generation Center (KGC):* KGC is used to generate and distribute keys for all the clients and the server. KGC also takes the role of group manager. It is responsible for group formation and dispute resolution.
- *Client:* Each client needs to register with KGC and joins a group in order to get services from the server. During the registration and join processes, each client computes or receives from KGC its group member id and group certificate. A legitimate client should be able to get services from the server without releasing its private information.
- *Data center:* The data center (may consist of cloud servers) is utilized to process the data for the server.
- *Server:* The server will get group information (group id and group public parameters) from KGC and will service the clients in the corresponding groups. It will generate monthly bills for the services provided to each client. The disputes between the server and a client is resolved by KGC.

B. Security Model

In our security model, we assume that KGC is a third-party. There is a secure channel between KGC and each client, which is used to transmit sensitive information during the registration and join operations. This secure channel is also used when disputes need to be resolved at KGC.

We consider that the following security goals need to be achieved:

- Privacy-preservation of client requests and identities. The server should be able to verify that a request is made by a legitimate client though the client identity is kept confidential; i.e, the server is able to tell that a request is made by a legal user without knowing who has made the request. Both the requests from clients and the responses from the server should be protected for confidentiality. The requests made by the same client within a billing period should be identifiable only by the server for billing purpose, but those from different billing periods should never be linked by the server or anybody else to protect the client privacy.
- Non-repudiation: If a client has made a request and serviced by the server, it cannot deny this activity in later time. If the server charges a client for a request that is not made by the client, the forged services should be identifiable.
- Accountability: The system is recordable and traceable. When there exists any security issue or a dispute between a client and the server, the system should be able to figure out who is responsible for what.

C. Preliminaries

1) Bilinear Maps: Bilinear maps is a cryptographic primitive that will serve as the basis for our proposed PPDIR scheme. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two bilinear groups of prime order q, and G be a generator of $\mathbb{G}_1$. Our proposed scheme makes use of a bilinear map: $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ with the following properties:

- *Bilinear:* A map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is bilinear if and only if for $\forall Q \in \mathbb{G}_1$ and $\forall a, b \in \mathbb{Z}_q^*$, we have $e(aG, bQ) = e(G, Q)^{ab}$. Here $\mathbb{Z}_q^* = \{1, \ldots, q-1\}$ is the Galois field of order q.
- *Non-degeneracy:* The generator G satisfies $e(G, Q) \neq 1$ for $\forall Q \in \mathbb{G}_1$.
- *Computability:* There is an efficient algorithm to compute $e(G, Q)$ for $\forall Q \in \mathbb{G}_1$.

Next we define the Gap Diffie-Hellman (DH) Group. To proceed, let's first introduce DLP, CDHP, and DDHP.

- *Discrete Logarithm Problem (DLP)*: Given a generator G and an element $Q \in \mathbb{G}_1$, DLP intends to find the integer $\alpha \in \mathbb{Z}_q$ such that $Q = \alpha G$ whenever such an integer exists.
- *Computation Diffie-Hellman Problem (CDHP)*: Given G, aG, and bG for $a, b \in \mathbb{Z}_q$, CDHP computes abG.
- Decision Diffie-Hellman Problem (DDHP): Given G, aG, bG, and cG for $a, b, c \in \mathbb{Z}_q$, DDHP decides whether $c \equiv ab \mod q$.

We call $\mathbb{G}_1$ a Gap DH Group if DDHP can be solved in polynomial time but there is no polynomial time algorithm to solve CDHP or DLP with a non-negligible probability. Such a group can be found in supersingular elliptic curve or hyper-elliptic curve over finite fields, and the bilinear pairing can be derived from the Weil or Tate parings [57], [58].

2) Group Signature: Generally speaking, a group signature scheme [59] consists of the following procedures:

- *Setup*: A group manager generates a group public key and a group secret key based on certain algorithms.
- *Join*: The group manager generates a secret key and a public key for a potential user, which might be a new group member.
- *Sign*: A group member signs its message using its secret key, member certificate, and so on.
- *Verify*: An algorithm exists to compute the validity of the signature based on the group public key.

- *Reveal*: Given a signed message and the group secret key, an algorithm (or the group manager) exists to return the identity of the signer.

A secure group signature must satisfy the following properties:

- *Correctness*: Signatures produced by a group member can be accepted by the verifier.
- *Unforgeability*: Only group members can sign messages on behalf of the group.
- *Anonymity*: Given a valid signature, it is computationally hard to identify the signer by anyone except the group manager.
- *Unlinkability*: Whether two different valid signatures are computed by the same group member is computationally hard to figure out by anyone except the group manager.
- *Exculpability*: Neither the group manager nor a group member can sign messages on behalf of other group members; a member is not responsible for any valid signature that is not produced by itself.

3) The Rabin Cryptosystem: The Rabin cryptosystem [60] is a provably secure public key encryption scheme. Its security strength is based on the practical difficulty of factoring the product of two large prime numbers.

- *Key Generation*: Each entity needs to create a public key n and the corresponding private key pair $\{n_1, n_2\}$ by finding two large random primes n_1 and n_2, with roughly the same size, and then setting $n = n_1 n_2$.
- *Encryption*: To send an encrypted message to *A*, *B* encrypts its plaintext M based on *A*'s public key n as follows: *B* treats M as an integer in the range $\{0, 1, 2, \ldots, n-1\}$ and then computes $C = M^2 \mod n$, where C is the ciphertext to be sent to *A*.
- *Decryption*: The receiver *A* recovers the plaintext M from C based on its private key pair $\{n_1, n_2\}$ according to the following procedure:
  1) *A* computes m_1, m_2, m_3, and m_4 based on the Chinese Remainder Theorem:
  $$\begin{aligned} m_1 &= C^{(n_1+1)/4} \mod n_1 \\ m_2 &= (n_1 - C^{(n_1+1)/4}) \mod n_1 \\ m_3 &= C^{(n_2+1)/4} \mod n_2 \\ m_4 &= (n_2 - C^{(n_2+1)/4}) \mod n_2 \end{aligned}$$
  2) *A* computes four messages M_1, M_2, M_3, and M_4 as follows:
  $$\begin{aligned} M_1 &= (am_1 + bm_3) \mod n \\ M_2 &= (am_1 + bm_4) \mod n \\ M_3 &= (am_2 + bm_3) \mod n \\ M_4 &= (am_2 + bm_4) \mod n \end{aligned}$$
  where $a = n_2(n_2^{-1} \mod n_1)$ and $b = n_1(n_1^{-1} \mod n_2)$.
  3) One of $\{M_1, M_2, M_3, M_4\}$ is the original message M. To determine which message is M, *B* can add predetermined redundancy before encryption, based on which *A* can figure out the original message M.

IV. The Proposed Scheme

In this section, we detail our scheme for Privacy Preservation and billing via Delayed Information Release (PPDIR).

TABLE I
The Notations and Their Descriptions Used in PPDIR

Notation	Description
H	Hash function
ID	unique client identity
s	master secret key
P	public key
k_j	session key
h_j	one-way hash chain

At the beginning of any billing period, each client needs to select a secret token and compute a one-way hash chain seeded by the token and $H(ID)$. The token will be securely released to the server at the end of the billing period to link all requests made by the same client during the current billing period for billing purpose. Each request is anonymous, is attached with one element in the hash chain, and is signed based on our group signature scheme such that the server can verify that the request is made by a valid member belonging to certain group but it cannot figure out the group member who has generated and signed the request. All the information transmitted from the client is protected by Rabin cryptosystem and the response from the server is encrypted by AES such that the computational complexity at the client side is low. Since the token will never be public, attackers cannot link the requests made by the same client within the same billing period; as the token is unique for each billing period, neither attackers nor the server can link the requests made at different billing periods. The notations used in this paper and their descriptions are shown in Table I.

The overall procedure of our proposed PPDIR scheme is summarized is Fig. 4. One can see that PPDIR mainly contains the following steps: **Setup, Registration and Join, Group Signature, Encryption, Decryption and Verification, Response and Delayed Information Release**. The details of each step are presented in the following subsections.

A. Setup

We assume that the server bootstraps the system. Specifically, the server first generates its private key $\{n_1, n_2\}$ and the corresponding public key $n = n_1 n_2$ based on the Rabin cryptosystem. The server keeps its private key $\{n_1, n_2\}$ and publishes its public key n.

KGC manages all the groups. To create a group, KGC needs to perform the following activities:

- *KGC* chooses a cyclic additive group $\mathbb{G}_1$ (which is a Gap DH group) of prime order q, which is generated by G, and a cyclic multiplicative group $\mathbb{G}_2$ with the same order q.
- *KGC* selects one secure symmetric encryption function such as *AES*, denoted by $\mathrm{Enc}()$, and three secure cryptographic hash functions $\{H, H_1, H_2\}$, where $H, H_1 : \{0,1\}^* \longrightarrow \mathbb{Z}_q^*$ and $H_2 : \{0,1\}^* \longrightarrow \mathbb{G}_1$, here $\mathbb{Z}_q^* = \{1, \ldots, q-1\}$.
- *KGC* chooses a random master key $s \in \mathbb{Z}_q^*$ and computes a public key $P = sG$.

KGC publishes the following public parameters for the newly-created group $\mathcal{G}$:

$$params = \{\mathbb{G}_1, \mathbb{G}_2, e, q, G, P, H, H_1, H_2\}. \quad (1)$$

where the bilinear pairing is a map $e : \mathbb{G}_1 \times \mathbb{G}_1 \longrightarrow \mathbb{G}_2$. *KGC* keeps the master secret key s.

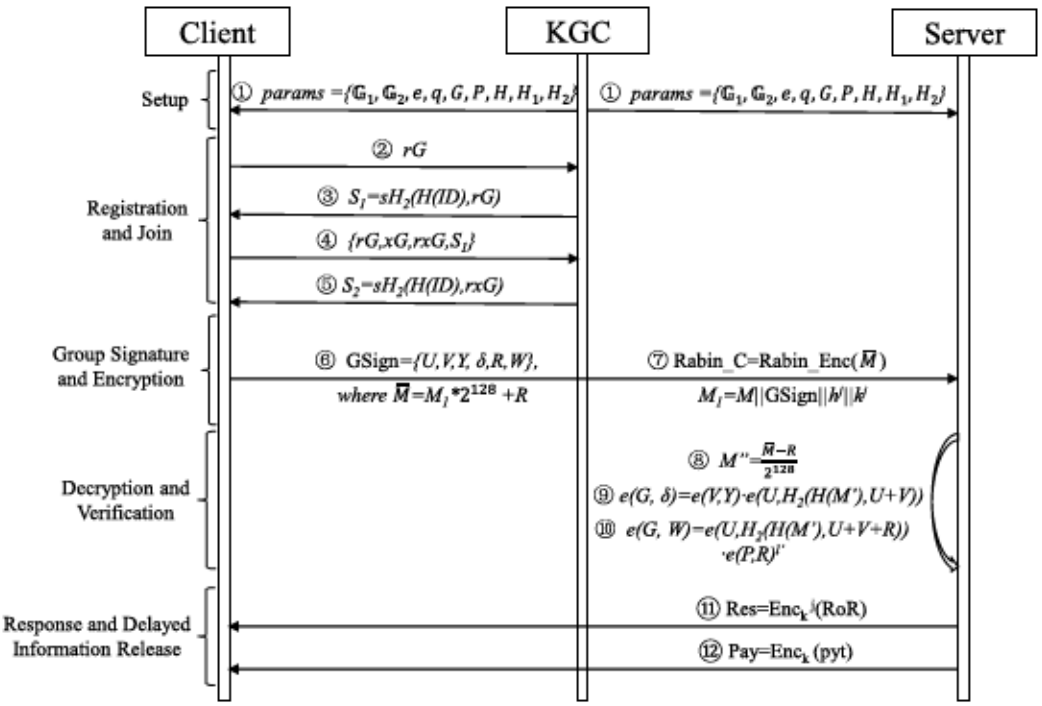


Fig. 4. The overall procedure of our PPDIR scheme.

B. Registration and Join

When a client with identity ID registers with KGC, the following two steps need to be performed:

1) The client randomly chooses an integer $r \in \mathbb{Z}_q^*$ as its private key, and sends rG to KGC via a secure channel.
2) KGC calculates $S_1 = sH_2(H(ID), rG)$, and sends it to the client via a secure channel.

The client's private key pair is $\{r, S_1\}$, and its public key is $H(ID)$. Note that we use $H(ID)$ instead of the ID itself as the public key of the client and the true ID of the client in our system is never released for privacy protection.

Here we assume a secure channel exists between KGC and each client such that the information exchanged during client registration can be protected. This assumption is reasonable as a client needs to be authenticated during registration in many practical applications that consider security a big issue.

After registration, the client can choose a group to join. Note that we separate the registration and join operations because a client may register with KGC but does not choose any group to join. The join process is outlined as follows:

1) The client chooses a random number $x \in Z_q^*$, and sends $\{rG, xG, rxG, S_1\}$ to KGC requesting to join the group $\mathcal{G}$ with public parameters $params = \{\mathbb{G}_1, \mathbb{G}_2, e, q, G, P, H, H_1, H_2\}$.
2) KGC verifies $e(rxG, G) \stackrel{?}{=} e(xG, rG)$ to check whether the client is a valid registered member of group $\mathcal{G}$. If true, KGC sends $S_2 = sH_2(H(ID), rxG)$ to the client via the secure channel; otherwise, the join request is declined.

Note that x is a private parameter selected by the client to bind itself with the group to join. The tuple $\{rG, xG, rxG, ID\}$ is used as the identity of the client in the group; thus it is added to the group member list. The client's member certificate is comprised of $\{S_2, rxG\}$ and its private signing key pair is $\{x, rx\}$.

A client does not need to register with the server as long as KGC reports to the server all the groups that may request services from the server. Each group has a unique identity $\mathcal{G}$ and its public parameters should be certified by KGC. Whenever a client sends an anonymous request to the server, the identity of the group it joins instead of its true identity should be included in order for the server to identify which group it belongs to while keeping its true identity confidential.

C. Group Signature

In [19], there are many pairing operations and exponential computation, the group signature is complexity and have a high computational overhead. So we propose a group signing procedure in this subsection in order for the server to ensure the authenticity of the client's request, i.e., to verify that the request is signed by a valid member of a group. Although our signature method is motivated by [18], it possesses a nice property that significantly differ itself from the one in [18]. Let M be the request message to be signed by a client. The following two steps outline the signing procedure.

1) The client randomly selects two numbers $k_1, k_2 \in \mathbb{Z}_q$;
2) The client employs its two signing keys $\{x, rx\}$ and the member certificate $\{rxG, S_2\}$ to sign the request M as follows:
   - $U = k_1 rxG$;
   - $V = (q - 1 - k_1)xG$;
   - $Y = rH_2(H(M), U + V)$;
   - $\delta = rxH_2(H(M), U + V)$;
   - $R = k_2 H_2(H(ID), rxG)$;
   - $l = H_1(H(M), U + V + R)$;
   - $W = lk_2 S_2 + k_1 rxH_2(H(M), U + V + R)$

The signature on M is the tuple $\mathsf{GSign} = \{U, V, Y, \delta, R, W\}$.

Our signature scheme proposed above allows a single signing key pair to be used to sign as many messages as a client would like to do because two random numbers $\{k_1, k_2\}$ are actually utilized to compute the signature for each message, while the one in [18], [19] requires a unique signing key pair for each message. This advantage can help conserve the storage space and decrease the overhead of managing a large number of signing key pairs, which is particularly useful when a client needs to sign many requests. In Subsection V-A, we will prove that our signature scheme still possesses the security properties required by group signatures.

D. Encryption

To preserve a client's privacy, we must prevent attackers from linking multiple transactions (each transaction consists of a client request and the corresponding response from the server) made by the same client within one billing period or across multiple billing periods. To achieve this objective, we employ hash chains and secret tokens to create different IDs for each transaction during each billing period. This method ensures that an attacker cannot trace a client even though it intercepts all the messages destined to or originated from the client. The basic idea is outlined as follows.

For each billing period T, we require the client to randomly choose a token τ and compute its initial seed h^0: $h^0 = H(H(ID) \\ ||\tau||T)$. A one-way hash chain $h^j = H(h^{j-1}, \tau)$ is also computed, with $j = 1, 2, \cdots$, and h^j to be used by the j^{th} request of the client in the current billing period T. Note that τ must be unique for each billing period but the client's identity ID does not need to be changed as it is never released.

1) For the j^{th} request M, the client chooses a session key k^j based on AES.
2) The client computes M_1 and R, where $M_1 = M||\mathsf{GSign}||h^j||k^j$, and R is the last 128-bits of M. Then the client computes $\overline{M} = M_1 * 2^{128} + R$. Note that R provides the required redundancy to facilitate the Rabin decryption.
3) The client employs the server's public key to encrypt $\overline{M}$ based on the Rabin encryption:

$$\mathsf{Rabin\_C} = \mathtt{Rabin\_Enc}(\overline{M}). \tag{2}$$

4) The client sends the ciphertext *Rabin_C* to the server.

This procedure provides a session key k^j for the j^{th} request in the current billing period, and k^j will be used by the server to encrypt the response based on AES if the client's request is validated. We choose AES to protect the session as it is strong enough while involving a low computational cost for both encryption and decryption.

E. Decryption and Verification

When the server receives Rabin_C, it decrypts the ciphertext using its private key (n_1, n_2), and verifies the group signature GSign. The decryption and verification procedure is outlined as follows:

1) The server decrypts Rabin_C and gets four decrypted messages $\{\overline{M_1'}, \overline{M_2'}, \overline{M_3'}, \overline{M_4'}\}$.
2) The server identifies the message $\overline{M}$ that possesses the redundancy (the R part). It then recovers the original message by computing $M'' = \frac{\overline{M}-R}{2^{128}}$, which consists of $\{M', GSign, h^j, k^j\}$
3) The server calculates $l' = H_1(H(M'), U + V + R)$.
4) The server verifies the following equations:

$$e(G, \delta) = e(V, Y) \cdot e(U, H_2(H(M'), U + V)) \tag{3}$$

$$e(G, W) = e(U, H_2(H(M'), U + V + R)) \cdot e(P, R)^{l'} \tag{4}$$

If (3) and (4) hold, the client's request is accepted by the server; otherwise, the server declines the request as the client is not verified.

Note that the server is able to verify whether the client is a legal registered user belonging to a certain group according to (3) and (4) but it does not know who requests the service. The security properties of this signature verification scheme is analyzed in Section V-A.

F. Response

After the server verifies that the client is legal, it sends a Response of Request (RoR) to the client:

1) The server creates its response, encrypts it with the session key k^j based on AES: $\mathsf{Res} = \mathtt{Enc}_{k^j}(\mathsf{RoR})$, and then sends Res to the client.
2) The server computes the payment b_j for the current service(s), and stores the request M, the signature GSign, the response RoR, the bill b_j, and the hash value h^j for billing purpose and accountability guarantee.

After receiving Res, the client decrypts it to obtain the response using session key k^j.

G. Delayed Information Release

At the end of the current billing period, the client is required to securely release the secret token τ and the hash chain seed h^0. This process is secured by the Rabin encryption based on the procedure outlined in Section IV-D. Let k be the session key selected by the client for information release and billing purpose.

When the server receives the released information, it decrypts to get the token τ and the session key k. Then it computes the hash chain according to the token τ and h^0. Based on the hash chain, the server can link all the requests made by the same client during the current billing period, and calculates the client's payments pyt. Then the server sends the bill to the client based on the following two steps:

1) The server encrypts the client's payment pyt using the client's session key k : $\mathsf{Pay} = \mathtt{Enc}_k(\mathsf{pyt})$.
2) The server sends Pay to the client.

When the client obtains Pay, it decrypts the message to obtain the payment pyt, and pay the bill. The client's detailed statement report can also be released, by following the same procedure.

V. Security Analysis

In this section, we investigate the security properties of our group signature scheme, and analyze the security strength of PPDIR in guaranteeing client privacy protection, non-repudiation, accountability, and authentication.

A. The Security Properties of Our Group Signature Scheme

In this section, we prove the security strength of the group signature scheme proposed in Section IV-C. Note that $\mathbb{G}_1$ is a Gap DH group in our scheme.

Theorem 1: No adversary can forge a valid signature with a non-negligible probability.

Proof: For contradiction we assume that an adversary can forge a valid signature. Then this signature must be able to pass the verification check established by (3) and (4). Let's consider (4). We notice that the right-hand side of (4) can be transformed to $e(U, H_2(H(M), U+V+R)) \cdot e(P,R)^l = e(U, H_2(H(M), U+V+R)) \cdot e(sG,R)^l$ while the left-hand side of (4) is $e(G,W)$. Comparing the two sides, one can see that if (4) is correctly established, the computation of W must involve s, which means that the attacker can obtain the master key s from the public key $P = sG$. This implies that the attacker can solve *DLP* in $\mathbb{G}_1$ by computing s from sG, which further implies that the attacker can solve the discrete logarithm problem in $\mathbb{G}_1$. This is impossible, and thus our assumption is not true. Therefore we conclude that no adversary can forge a valid signature with a non-negligible probability. □

Theorem 2: No group member (client) can impersonate and sign on behalf of any other group member with a non-negligible probability.

Proof: Consider the following impersonation attacks:

Case 1: The attacker does not collude with KGC.

Assume that a client with identity ID_j is the attacker who intends to impersonate another client with identity ID_i to sign ID_i's request M. In order to forge a valid signature for M that can pass the verification check, the attacker needs to choose appropriate values of U, V, Y, δ, R, W such that (3) and (4) can hold. Because G and P are public, the attacker must establish a valid relationship between $\{G,P\}$ and $\{U,V,Y,\delta,R,W\}$. Considering the two side of (3) and (4), one can see that it is easy for the attacker to forge a signature when $U=V=G$ or $U=V=P$, which reduces the difficulty to establish a valid relationship between $\{G,P\}$ and $\{U,V,Y,\delta,R,W\}$ as the attacker only needs to consider the relationship between $\{G,P\}$ and $\{Y,\delta,R,W\}$.

- When the attacker chooses $U=V=G$:
  The right-hand side of (3) can be transformed to:

$$\begin{aligned} &e(V,Y)\cdot e(U,H_2(H(M),U+V)) \\ &\quad = e(G,Y)\cdot e(G,H_2(H(M),U+V)) \\ &\quad = e(G,Y+H_2(H(M),U+V)) \end{aligned} \tag{5}$$

  Comparing the left-hand side of (3) and the right-hand side of (5), one can see that the attacker can pass the verification check (3) when $\delta = Y + H_2(H(M),U+V)$, where Y can be selected randomly by the attacker.
  But it is not easy for the forged signature to pass the verification check (4), as the attacker does not know the certificate component S_2 of ID_i. The right-hand side of (4) can be equivalently transformed to:

$$\begin{aligned} &e(U,H_2(H(M),U+V))\cdot e(P,R)^l \\ &\quad = e(G,H_2(H(M),U+V))\cdot e(sG,R)^l \\ &\quad = e(G,H_2(H(M),U+V))\cdot e(G,lsR) \\ &\quad = e(G,lsR+H_2(H(M),U+V)) \end{aligned} \tag{6}$$

  Comparing the left-hand side of (4) and the right-hand side of (6), one can see that the forged signature can pass the verification check (4) when $W = lsR + H_2(H(M),U+V)$. Because the attacker does not know the master key s, which is held by KGC, the attacker can only brute-force W to establish $W = lsR + H_2(H(M),U+V)$, which can succeed with a probability $1/(q-1)$.

- When the attacker chooses $U=V=P$:
  The right-hand side of (3) can be equivalently transformed to

$$\begin{aligned} &e(V,Y)\cdot e(U,H_2(H(M),U+V)) \\ &\quad = e(sG,Y)\cdot e(sG,H_2(H(M),U+V)) \\ &\quad = e(G,s(Y+H_2(H(M),U+V))) \end{aligned} \tag{7}$$

  Comparing the left-hand side of (3) and the right-hand side of (7), one can see that the forged signature can pass the verification check (3) only when $\delta = s(Y + H_2(H(M),U+V))$. However, the attacker cannot compute $\delta = s(Y+H_2(H(M),U+V))$ because it does not know the master key s. Brute-forcing all possible values of δ has a success probability of only $1/(q-1)$ to establish $\delta = s(Y+H_2(H(M),U+V))$.
  Now we consider whether the attacker can pass the verification check (4). The right-hand side of (4) can be transformed to

$$\begin{aligned} &e(U,H_2(H(M),U+V))\cdot e(P,R)^l \\ &\quad = e(sG,H_2(H(M),U+V))\cdot e(sG,R)^l \\ &\quad = e(G,sH_2(H(M),U+V))\cdot e(G,lsR) \\ &\quad = e(G,s(lR+H_2(H(M),U+V))) \end{aligned} \tag{8}$$

  Comparing the left-hand side of (4) and the right-hand side of (8), one can see that the forged signature can pass (4) only when $W = s(lR+H_2(H(M),U+V))$. Because the attacker does not know the master key s, it can not establish $W = lsR + H_2(H(M),U+V)$; Brute-forcing all possible values of W has a success probability of only $1/(q-1)$.
  Therefore we claim that in this case the attacker can forge a valid group signature with a probability at most $1/(q-1)^2$ as (3) and (4) must simultaneously hold.

According to the above analysis, one can see that the attacker can impersonate a valid group signature with a probability at most $1/(q-1)$ as it needs to brute-force all possible values of δ and W. Recall from Section IV-A that q is a very large prime number; therefore the probability of successfully impersonating a legal group member is negligible.

Case 2: The attacker colludes with KGC.

In this case, the attacker with identity ID_j can obtain $\{r_iG, x_iG\}$ and the member certificate $\{S_2, r_ix_iG\}$ of an honest group member with identity ID_i by colluding with KGC. The attacker chooses $k_1', k_2' \in \mathbb{Z}_q$, $r_x, x_x \in \mathbb{Z}_q^*$, and obtains the following values:

$$\begin{aligned} U &= k_1' r_i x_i G \\ V &= (q-1-k_1')x_iG \\ Y &= r_x H_2(H(M),U+V) \\ \delta &= r_x x_x H_2(H(M),U+V) \\ R &= k_2' H_2(H(ID_i), r_ix_iG) \\ l &= H_1(H(M),U+V+R) \\ W &= lk_2'S_2 + k_1' r_x x_x H_2(H(M),U+V+R) \end{aligned}$$

According to the (3), we have:

$$\begin{aligned} e(G,\delta) &= e(G, r_x x_x H_2(H(M),U+V)) \\ &= e(r_x x_x G, H_2(H(M),U+V)) \end{aligned} \tag{9}$$

On the other hand, the right-hand side of (3) can be equivalently transformed to

$$\begin{aligned} & e(V,Y)\cdot e(U,H_2(H(M),U+V)) \\ &= e((q-1-k_1')x_tG, r_xH_2(H(M),U+V))\cdot e(k_1'r_tx_tG, \\ &\quad H_2(H(M),U+V)) \\ &= e((q-1-k_1')r_xx_tG, H_2(H(M),U+V))\cdot e(k_1'r_tx_tG, \\ &\quad H_2(H(M),U+V)) \\ &= e(((q-1-k_1')r_xx_t+k_1'r_tx_t)G, \\ &\quad H_2(H(M),U+V)) \end{aligned} \tag{10}$$

Comparing (9) and (10), we observe that (3) is established only when $r_xx_t = r_tx_t$ holds. The probability that $r_xx_t = r_tx_t \mod q$ holds is $1/(q-1)$, which is very low because q is very large.

Similarly, according to (4), we have:

$$\begin{aligned} e(G,W) &= e(G, lk_2'S_2 + k_1'r_xx_xH_2(H(M'),U+V+R)) \\ &= e(G,k_2'S_2)^l \\ &\quad \cdot e(k_1'r_xx_xG, H_2(H(M),U+V+R)) \end{aligned} \tag{11}$$

On the other hand, the right-hand side of (4) can be equivalently transferred to

$$\begin{aligned} & e(U,H_2(H(M),U+V+R))\cdot e(P,R)^l \\ &\quad = e(k_1'r_tx_tG, H_2(H(M),U+V+R))\cdot e(sG, \\ &\quad k_2'H_2(H(ID_t),r_tx_tG))^l \\ &\quad = e(k_1'r_tx_tG, H_2(H(M),U+V+R))\cdot e(G, \\ &\quad k_2'sH_2(H(ID_t),r_tx_tG))^l \\ &\quad = e(k_1'r_tx_tG, H_2(H(M),U+V+R)) \\ &\quad \cdot e(G,k_2'S_2)^l \end{aligned} \tag{12}$$

Comparing (11) and (12), one can say that in order for (4) to hold, we need

$$\begin{aligned} & e(k_1'r_xx_xG, H_2(H(M),U+V+R)) \\ &\quad = e(U, H_2(H(M),U+V+R))) \\ &\quad = e(k_1'r_tx_tG, H_2(H(M),U+V+R))) \end{aligned}$$

which requires that

$$r_xx_xG = r_tx_tG.$$

The probability that $r_xx_x = r_tx_t \mod q$ holds is $1/(q-1)$, which is very low because q is very large. If the attacker wants to choose r_t and x_t from $r_xx_xG = r_tx_tG$, it has to solve a *DLP* in $\mathbb{G}_1$.

Obviously, if the attacker colludes with KGC to forge a valid signature, it has to pass the verification (3) and (4); and its success probability is $1/(q-1)^2$. □

Theorem 3: KGC cannot impersonate and sign on behalf of a group member with a non-negligible probability under the assumption of the hardness of DLP in $\mathbb{G}_1$.

The proof is omitted here as it is similar to the **Case 2** in Theorem 2.

Theorem 4: The two verification equations (3) and (4) hold if and only if the signer is a legal group member.

Proof: If the signer is a legitimate group member, we can prove that (3) and (4) are established as follows. Since the values $\{U, V, Y, \delta, R, W\}$ come from a legitimate group member, the left-hand side of (3) can be equivalently transformed to

$$\begin{aligned} e(G,\delta) &= e(G, rxH_2(H(M),U+V)) \\ &= e(rxG, H_2(H(M),U+V)) \end{aligned} \tag{13}$$

and the right-hand side of (3) can be equivalently transformed to

$$\begin{aligned} & e(V,Y)\cdot e(U,H_2(H(M'),U+V)) \\ &\quad = e((q-1-k_1)xG, rH_2(H(M),U+V))\cdot e(k_1rxG, \\ &\quad H_2(H(M'),U+V)) \\ &\quad = e((q-1-k_1)rxG, H_2(H(M),U+V))\cdot e(k_1rxG, \\ &\quad H_2(H(M'),U+V)) \end{aligned} \tag{14}$$

Comparing (13) and (14), one can see that (3) is established if $M' = M$ holds:

$$\begin{aligned} & e((q-1-k_1)xrG, H_2(H(M),U+V))\cdot e(k_1rxG, \\ &\quad H_2(H(M'),U+V)) \\ &\quad = e((q-1-k_1)xrG, H_2(H(M),U+V))\cdot e(k_1rxG, \\ &\quad H_2(H(M),U+V)) \\ &\quad = e(rxG, H_2(H(M),U+V)) \\ &\quad = e(G, rxH_2(H(M),U+V)) \\ &\quad = e(G,\delta) \end{aligned}$$

Obviously $M' = M$ holds as the request M and its signature are protected by the sever's public key, assuming that no random errors from the communication channel is introduced. Therefore (3) is established.

Similarly, the left-hand side of (4) can be equivalently transformed to

$$e(G,W) = e(G, lk_2S_2 + k_1rxH_2(H(M),U+V+R)), \tag{15}$$

and the right-hand side of (4) can be equivalently transferred to

$$\begin{aligned} & e(U,H_2(H(M'),U+V+R))\cdot e(P,R)^{l'} \\ &\quad = e(k_1rxG, H_2(H(M'),U+V+R))\cdot e(sG, \\ &\quad k_2H_2(H(ID),rxG))^{l'}, \\ &\quad = e(G, k_1rxH_2(H(M'),U+V+R))\cdot e(G, \\ &\quad l'k_2sH_2(H(ID),rxG)) \\ &\quad = e(G, k_1rxH_2(H(M'),U+V+R))\cdot e(G, l'k_2S_2) \\ &\quad = e(G, l'k_2S_2 + k_1rxH_2(H(M'),U+V+R)) \end{aligned} \tag{16}$$

Comparing (15) and (16), we notice that if $M' = M$ and $l' = l$ hold, (4) is established. Obviously, $M' = M$ and $l' = l$ hold when there is no random errors introduced by the imperfect communication channel.

On the other hand, if (3) and (4) are established, the valid signature must be produced by a signer who is a legitimate group member, according to Theorem 1, Theorem 2, and Theorem 3. Note that this verification procedure allows the server to verify whether a client is a legal group member without disclosing who the client is. □

From the theorems established above, it is easy to deduce that our group signature scheme possesses the following security properties: unforgeability, anonymity, unlinkability, and exculpability.

B. Protection on Client Identity and Requests

In this section, we present how our proposed scheme PPDIR can provide protection on client identity and client requests during a billing period and across multiple billing periods. Recall that a client needs to release to the server its token and initial seed at the end of each billing period for billing purpose and the released information is protected by Rabin encryption via the server's public key. Therefore there is no way for an adversary (other than the server) to get any information regarding the client's identity and requests. Thus in this section we focus on the protection of client identity and requests at the server side.

1) Protecting Client Identity: A client hides its identity as follows:

- Its identity ID is hidden via the hash function H: $H(ID)$, and $H(ID)$ is never released to the public (the server, other clients, adversaries). The true ID is known to KGC only as the client needs to authenticate itself to KGC for registration purpose. KGC is assumed to be trusted.
- When a request is received, the server only needs to verify whether the request is made by a member from a legal group via group signature verification; the identify of the client who made the request is not needed during a billing period or across multiple billing periods.
- The server can not figure out the client's identity based on the initial seed h^0, even though h^0 is related to the client's identity. This is because the client's identity is protected by the one-way hash function H. On the other hand, since the initial seed of each billing period is different, the server can not link together a client's tokens and hash values generated for different billing periods to reveal the client's identity.

2) Protecting Client Requests: We will use Lemma 5 to demonstrate how client requests are protected.

Lemma 5: The server can not link the requests made by a client in a billing period before the client releases its token and initial seed at the end of the billing period, and can not link the requests made by a client across multiple billing periods even though the client releases its token and initial seed at the end of each billing period.

Proof: In our scheme, a client is requested to release its token τ and initial seed h^0 at the end of each billing period T, in order to compute the monthly bill for the client. Our scheme also requires that the token for each billing period must be randomly selected and must be unique, which implies that the initial seed for each billing period must be unpredictable and must be unique, as $h^0 = H(H(ID)||\tau||T)$.

Within a billing period the server can not link the requests made by the same client before the client releases its token τ and initial seed h^0 because the client's requests are linked by a one-way hash chain seeded by h^0, which is hidden and can be computed only when both h^0 and τ are available.

The server can not link the requests made by the same client at different billing periods because the token is updated for each billing period by the client, and the token and initial seed for different billing periods are different. There is no linkage between two requests made at different billing periods. Thus there is no way for the sever to get sufficient information to link the client's requests from different billing periods.

In summary, our PPDIR scheme ensures that the server can not link the requests made by the same client within a billing period before the client releases its token and initial seed at the end of the billing period, and can never link the requests made by the same client across different billing periods. □

The analysis made above indicates that our PPDIR scheme obfuscates the linkability of the client's requests within a billing period and thus the server can not link two requests made by the same client before the client releases its token and initial seed. Additionally, the client updates its token τ at the beginning of each billing period to ensure that the initial seed h^0 for each billing period is unique. This strategy destroys the linkage of the client's requests across multiple billing periods. As all released information and the client requests are protected by the Rabin cryptosystem, an adversary can not get any information regarding the client identify and requests.

C. Non-Repudiation and Accountability

1) Non-Repudiation: In this section, we prove that our proposed PPDIR scheme has the property of non-repudiation for both the client and the server.

Theorem 6: If a client makes a request and obtains the service from the server, it can not deny making the request and obtaining the service from the server in a later time.

Proof: If a client follows the PPDIR protocol to make requests and get services, it is easy to reveal its identity if it intends to deny its requests at a later time as the signature is impossible to forge and the disputes can be easily resolved at KGC with full information (token and initial seed). This case is ignored here.

If a client does not follow the protocol, there might be multiple possibilities for it to deny receiving the service from the server. For example, it may not release its token τ and h^0 to the server, or it may only release partial information (the token τ, or h^0, or the j^{th} request's hash value h^j) to the server. Under these scenarios, PPDIR can still reveal the client's identity to prevent such illegal activity. We discuss these issues by considering the following four cases:

Case 1: A client releases neither its token τ nor its h^0.

If a client does not release its token τ and h^0, the server will send all the hash values and signatures corresponding to the unpaid services to KGC. KGC can figure out the identity of the client as follows:

First, KGC searches its database to identify the suspected client by checking whether (17) holds for any client in its database, where U and V are obtained from an unpaid request.

$$e(U, G) \cdot e(V, rG) = e(r x G, G) \tag{17}$$

In order to verify whether a suspected client does have a cheating action, KGC verifies whether the signature belongs to the client:

$$e(G, S_2) = e(P, H_2(H(ID), r x G)) \tag{18}$$

$$e(G, S_1) = e(P, H_2(H(ID), rG)) \tag{19}$$

If (18) and (19) are established, the client can not disown its signature as $\{U, V, rxG, rG\}$ confirms the identity of the client. Therefore a client can not deny a request it has made by releasing neither τ nor h^0.

Case 2: A client releases only its token τ.

If a client releases its token τ only, the server can compute the hash chain according to $h^j = H(h^{j-1}, \tau)$ from all the unpaid services (the hash value is stored for each serviced request) and τ to retrieve all the requests made by the client, and link them to the client's monthly bill. The drawback for

the server is that it has to spend more time to search the client's requests. However, this should not be a big concern as the server can be supported by data centers with high computational capacity.

Case 3: A client releases its initial seed h^0 or the j^{th} hash value h^j but not the token τ.

If a client releases only its initial seed h^0 or the j^{th} hash value h^j, the server can not retrieve the requests made by the client based on h^0 or h^j; thus it will send all the hash values and signatures corresponding to the unpaid services to KGC, which can figure out the identity of the client as in **Case 1**.

Case 4: A client releases its token τ and the j^{th} hash value h^j.

If a client releases its token τ and a hash value h^j other than h^0, the server can retrieve the j-th request and the following ones via $h^j = H(h^{j-1}, \tau)$. For the first $(j-1)$ requests, the server can link them as in **Case 2**.

According to the above analysis, we conclude that no client can deny receiving the service to avoid paying the bill if it did make the request. □

Note that in **Case 1** KGC needs to verify (17) for 50% of its registered clients in the same group on average. This computational overhead can be supported by data centers. On the other hand, it is also manageable for KGC to handle the computational load caused by resolving disputes between clients and the server, as in practice there exists only a small percentage of dishonest clients that do not obey the protocol requirements.

Theorem 7: The server can not forge any client's request.

Proof: If a client does not request a service, but the server asks the client to pay for the service, the dispute needs to be resolved at KGC. KGC requests from the server the GSign corresponding to the controversial service and from the client its group member certificate and group member id. With such information KGC can figure out whether the GSign provided by the server is made by the client. If not, it is obvious that the client did not make the request.

Notice that the server can not forge a request even after obtaining the token and initial seed from the client as the group signature GSign is unforgeable according to Theorems 1 and 2. The server can not ask the client to pay a service made in previous request as the token needs to be changed for each billing period. This also implies that our PPDIR scheme can resist replay attacks. □

2) Accountability: Accountability is required to secure a system from the aspects of integrity, confidentiality, and privacy [61]–[65]. An accountability mechanism is typically utilized to figure out who is responsible for what. In essence, accountability means that the system is recordable and traceable, which implies that making any entity in the system accountable for all its actions. Under such a consideration, our PPDIR scheme is accountable as the information stored for each request at the server can be used as an evidence for dispute resolution, as indicated by Theorems 6 and 7; therefore no one can deny its actions. Thus we claim that PPDIR has the property of accountability.

VI. Performance Evaluation

In this section, we evaluate the computational and communication overheads of PPDIR for both the client and the server.

TABLE II

The Operations of KGC, One Client (One Request), and the Server (One Response)

Steps	KGC	One Client	the server
(a)	C_m	0	0
(b)	$2 * C_m + 2 * C_p$	$3 * C_m$	0
(c)	0	$7 * C_m$	$6 * C_p + Rabin_d$
(d)	0	$7 * C_m$	$6 * C_p + Rabin_d$

Notes: (1) (a): Setup; (b): Registration and Join; (c): Encryption, Decryption, Response; and (d): Information Release. (2) C_m is the computational cost of a multiplication operation in $\mathbb{G}_1$. (3) C_p is the computational cost of a pairing operation. (4) $Rabin_d$ is the computational cost of Rabin Decryption.

A. Computational Overhead

Consider the following two sets of operations: the first set contains the exponentiatial operation in $\mathbb{G}_1$, pairing, and Rabin decryption $Rabin_d$, and the second set includes AES encryption/decryption, Rabin encryption (just one modular square operation), and hash operations. The computational cost of the second set is negligible compared to that of the first set [66]. On the other hand, because KGC rarely uses decryption (decryption is used by KGC to resolve disputes between clients and the server only), we can neglect KGC's verification operation.

Table II summarizes the operations of KGC, one client, and the server. In this table, we denote the computational cost of a multiplication operation in $\mathbb{G}_1$ as C_m, and a pairing operation as C_p.

Let's first consider the computational overhead in KGC. As described in Setup (Section IV-A), KGC needs to perform one multiplication operation in $\mathbb{G}_1$: $P = sG$, which is computed once only. In Registration and Join (Section IV-B), KGC has two multiplication operations in $\mathbb{G}_1$: $\{S_1, S_2\}$, and two pairing operations $\{e(rxG, G), e(xG, rG)\}$. The combined overhead is thus $2 * C_m + 2 * C_p$.

At the client side, no computation is needed for setup. The computational cost for the registration and join process is $3 * C_m$, which occurs only once. In group signature, encryption, decryption, and response processes (Section IV-C, IV-D, IV-E, and IV-F), a client needs to perform seven multiplication operations $\{U, V, Y, \delta, R, lk_2S_2, k_1rxH_2(H(M||T), U+V+R)\}$ for each request. Thus the computational cost is $7 * C_m$ for a client considering one request. Similarly, there are seven multiplication operations during the information release process for each billing period (Section IV-G).

At the server side, the computational cost occurs in the decryption and response processes (Section IV-E and IV-F). the server involves six pairing operations (Equation (3) and (4)) and one Rabin decryption operation $Rabin_d$ for each response. The combined overhead is thus $6 * C_p + Rabin_d$ for each response. The same amount of operations are needed during the information release phase (Section IV-G) for each billing period.

Table III summarizes the computational cost of N clients, with each making n_r requests in a billing period. Because each client just needs to perform registration and join operation once to obtain its signing key and certificate key, which are used by all requests, the overhead of registration and join can be negligible. The total computational costs of KGC,

TABLE III
COMPUTATIONAL OVERHEAD FOR A BILLING PERIOD

	KGC	Clients	the server
Setup	C_m	0	0
Registration and Jion	$(2 * C_m + 2 * C_p) * N$	$3 * C_m * N$	0
Billing period	0	$((7 * C_m) * n_r + 7 * C_m) * N$	$(6 * (C_p * n_r + C_p) + Rabin_d * (1 + n_r)) * N$

Notes: (1) N is the number of group members; n_r is the number of requests made by the client in a billing period. (2) The computational cost of a multiplication operation in $\mathbb{G}_1$ and a pairing operation are denoted by C_m and C_p, respectively.

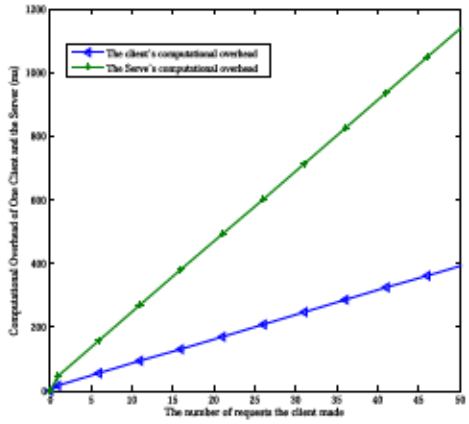


Fig. 5. The computational overhead of one client and the server in a billing period.

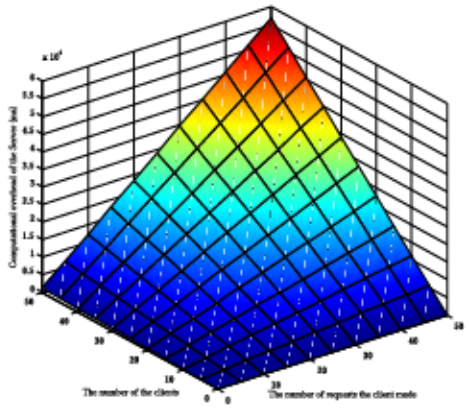


Fig. 6. The Computational overhead of the clients in a group within one billing period.

the clients, and the server are respectively $(2 * C_m + 2 * C_p) * N + C_m$, $((7 * C_m) * n_r + 7 * C_m) * N$, and $(6 * C_p + Rabin_d) * n_r * N + (6 * C_p + Rabin_d) * N$, for N clients with each making n_r requests during one billing period.

We also conduct experiments on a 3.0GHz-processor, 2GB memory computing machine to record the computational cost of major operations. Our results indicate that for $\mathbb{G}_1$ over the Freeman-Scott-Teske (FST) curve [67], a single multiplication operation costs 1.1 ms and the corresponding pairing operation costs 3.1ms; meantime, a 2048-Rabin decryption costs 3.71ms.

Fig. 5 illustrates the computational overhead of one client and the server. The client has a much lower computational overhead as it involves only multiplication operations, which are faster than the pairing operations. Fig. 6 shows the computational overhead of a group of clients. The computational overhead increases with the number of the clients and the number of requests each client makes.

B. Communication Overhead

We analyze the communication overhead of the proposed PPDIR scheme in terms of the public key size and the ciphertext size. Since most pairing-based cryptosystems need to work in a subgroup of an elliptic curve by representing the elliptic curve points using point compression [68], we choose $\mathbb{G}_1$ with 160-bit order. The lengths of the elements in $\mathbb{G}_1$ and $\mathbb{G}_2$ are roughly 161-bit and 1024-bit, respectively. Assume that the length of a hash function is 160-bit. The server needs 2048-bit to cover the length of the client's requests as each contains a number of elements including the request message, the signature, the session key, etc.

From the Setup phase in Section IV-A, we know that the public key is a constant. The size of the public key can be computed from **Equation** (20). The size of the system parameters *params* is $160 + 1024 + 161 * 3 + 160 * 3 + 2048 =$ 4195-bit $=$ 525 bytes, which is computed by (20).

$$PK_L = |\mathbb{G}_1| + |\mathbb{G}_2| + |q| + |G| + |P| + 3|H| + |n| \quad (20)$$

In Section IV-B, we notice that the client executes the Registration and Join operation only once. The size of the total exchanged messages in Registration and Join is $|rxG| + |xG| + 2|rG| + |ID| + 2|S_1| + |S_2| = 322 + 966 = 1288$-bit, which can be explained as follow: if the client does not want to update its private key and signing key, it can continuously use its previous private key and signing key; for Registration, the client sends rG to the server, and the server responses S_1 to the client; thus the length of the exchanged messages is $|rG| + |S_1|$, which is $161 + 161 = 322$-bit; for the Join operation, the client sends the message $\{rxG, xG, rG, ID, S_1\}$ to KGC, and KGC replies $S_2 = sH_2(H(ID)||T, rxG)$ to the client; thus the length of the exchanged message is $|rxG| + |xG| + |rG| + |ID| + |S_1| + |S_2|$, which is $161 * 3 + 161 * 3 = 966$-bit.

In the Group Signature and Encryption phases, the client encrypts the signature and the original request message via Rabin Encryption; thus the length of the signed and encrypted message equals the length of the public key n, which is 2048-bit. At the server side, the server encrypts the response RoR via AES block cipher, which needs one 256-bit block AES. At the end of the current billing period, the client releases its token τ and the initial seed h^0 to the server, and the length of the encrypted released message is 2048-bit. When the server obtains the token τ and the initial seed h^0, it needs to compute the bill, and sends the bill to the clients. The server needs one block AES (256 bit) to encrypt the bill. Thus the total message size is $2048 + 256 = 2304$-bit $= 288$ bytes.

In Table IV, we assume that there are N clients in the group. Each client makes n_r requests with the server during one billing period. Note that a client only needs to run Registration and Join Protocol once to register into the system. The communication overhead of this phase is $4195 + 1288 = 5483$-bit, which is about 686 bytes. During one billing period, the total

TABLE IV
COMMUNICATION OVERHEAD OF ONE CLIENT AND A GROUP OF CLIENTS

	One Client (bytes)	Group Clients (bytes)
Setup, Registration &Join	686	$686 * N$
During one billing period	$288 * m + 288$	$(288 * m + 288)N$

Note: N is the number of group members; m is the number of requests made by the client in one billing period.

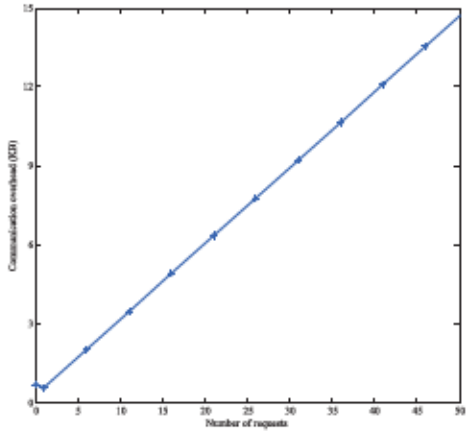


Fig. 7. The communication overhead of one client in a billing period.

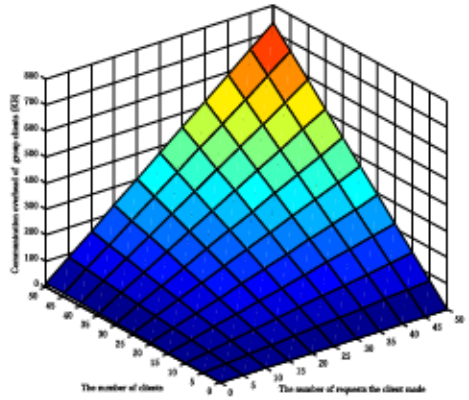


Fig. 8. The communication overhead of a group of clients in a billing period.

communication overhead for a client is $(2304 * n_r + 2304)$ bits $= (288 * n_r + 288)$ bytes. For a group of N clients, the communication overhead to register and join the group is $686 * N$ bytes, which can be ignored as the Registration and Join operation is performed only once for each client. Therefore the total communication overhead of the group during one billing period is $(288 * n_r + 288) * N$.

Fig. 7 demonstrates the communication overhead of one client in a billing period. One can see that the communication overhead increases linearly with the number of requests made. Fig. 8 illustrates the communication overhead of a group of clients. We observe that the communication overhead increases with the number of clients and the number of requests made by each client.

VII. DISCUSSIONS AND FUTURE RESEARCH

A. Cons and Pros of PPDIR

PPDIR makes use of the following innovative designs: (1) a novel lightly-weighted group signature scheme that supports anonymous communications and signature verifications, in which one signing key pair can be used to sign as many requests as the client wishes while traditional group signature schemes require one signing key pair per message; (2) a one-way hash chain seeded by a token to assist the server for linking the requests made by the same client within one billing period but not those in different billing period such that traditional billing can be implemented while client activities in different billing period cannot be correlated; (3) a Rabin cryptosystem based scheme such that client requests can be protected by Rabin encryption which takes only one module multiplication while the more expensive decryption operations are performed at the server side, enabling the applications of PPDIR in low-end client devices. With such a design, PPDIR can be potentially applied in many different domains, as discussed in next subsection.

PPDIR represents our exploratory effort towards privacy protection while supporting traditional billing in typical client-server based applications. It cannot completely prevent the server from getting client information as bills need to be computed by the server but the information is disclosed only to the server when needed via delayed information release. If the server is not allowed to get any information while traditional billing still needs to be realized, advanced crypto schemes such as homomorphic encryption need to be investigated, which could place high computational overhead at the client side. On the other hand, PPDIR does not consider privacy leakage in prepaid environments, and it assumes the existence of a trusted third party, which makes it hard to be applied in distributed environments.

B. Potential Applications

We describe two applications, i.e., location based services and smart metering, in the Introduction section to motivate the design of PPDIR. These two applications share the following properties: 1) the client requests should be kept private; 2) traditional billing in which the server computes a bill for each client at the end of each billing period based on the amount of services offered to the client is usually preferred. Nevertheless, there exists a dilemma between privacy protection and billing: the former requires all client information to be kept confidential while the latter needs all information to be disclosed to the server. PPDIR solves the problem with its novel designs: a new group signature scheme allowing the server to instantly serve a legit but anonymous client, a hash chain permitting the client to link all requests of a client within one billing period but not the requests from different billing period, and the Rabin encryption scheme shifting the heavy computation overhead from the lightly-weighted client devices to the server side. With PPDIR, one can achieve both privacy protection and billing, as the server can only compute the bill for a legit client at the end of a billing period but cannot link the client activities in different billing period.

There exist many other applications that can benefit from PPDIR. Here are two examples. Electric vehicles (EV) are usually charged when parked at public garages and many EV charging services require users to subscribe and pay periodically. If the EV charging information is not kept secret, the driver can be traced and its activities can be inferred. With PPDIR, the driver can be protected while paying bills regularly, i.e., at the end of a billing period, the server is allowed to compute a bill for the EV but it cannot link the activities of the driver from different billing periods. Credit card payments at point-of-sale can release the card holder's critical information such as preferences and behaviors if no proper measures are taken. With PPDIR, the card holder's information can be protected while payment can be correctly computed without much hassle.

C. Future Research

In our future research, we will proceed along the following two directions. First, we intend to instantiate our PPDIR scheme for specific applications that require not only fine-grained privacy-preserving data collection but also other application-specific demands such as balance checking in credit card payment, for the purpose of reducing the computational overhead and communication overhead at the client side. Second, we plan to consider the properties of secure multiparty computation, based on which one can design privacy-preserving schemes without a trusted third party such that the proposed scheme can be more suitable for practical applications.

VIII. Conclusion

In this paper, we first propose a new group signature mechanism to meet the requirements of our PPDIR scheme, and then present a privacy preservation scheme via delayed information release to protect applications that take the client-server architecture and involve billing services. Our scheme explores efficient cryptographic primitives to achieve privacy preservation. We also prove the security properties of the proposed group signature scheme and analyze the security strength of our PPDIR scheme by investigating how PPDIR can achieve privacy-preservation, non-repudiation, accountability, and so on. We evaluate the performance of PPDIR in terms of communication overhead and computational overhead.

References

[1] C. Y. T. Ma, D. K. Y. Yau, N. K. Yip, and N. S. V. Rao, "Privacy vulnerability of published anonymous mobility traces," *IEEE/ACM Trans. Netw.*, vol. 21, no. 3, pp. 720–733, Jun. 2013.

[2] Y.-A. de Montjoye, C. A. Hidalgo, M. Verleysen, and V. D. Blondel, "Unique in the crowd: The privacy bounds of human mobility," *Sci. Rep.*, vol. 3, no. 1, pp. 1–5, Dec. 2013.

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

[4] Y. Li, W. Dai, Z. Ming, and M. Qiu, "Privacy protection for preventing data over-collection in smart city," *IEEE Trans. Comput.*, vol. 65, no. 5, pp. 1339–1350, May 2016.

[5] K. Xing, C. Hu, J. Yu, X. Cheng, and F. Zhang, "Mutual privacy preserving k-means clustering in social participatory sensing," *IEEE Trans. Ind. Informat.*, vol. 13, no. 4, pp. 2066–2076, Aug. 2017.

[6] R. Anderson and S. Fuloria, "On the security economics of electricity metering," in *Proc. WEIS*, 2010, pp. 1–18.

[7] F. D. Garcia and B. Jacobs, "Privacy-friendly energy-metering via homomorphic encryption," in *Security and Trust Management*. Berlin, Germany: Springer, 2011, pp. 226–238.

[8] C. Hu, W. Li, X. Cheng, J. Yu, S. Wang, and R. Bie, "A secure and verifiable access control scheme for big data storage in clouds," *IEEE Trans. Big Data*, vol. 4, no. 3, pp. 341–355, Sep. 2018.

[9] R. Paulet, M. G. Kaosar, X. Yi, and E. Bertino, "Privacy-preserving and content-protecting location based queries," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 5, pp. 1200–1210, May 2014.

[10] X. Huang *et al.*, "Cost-effective authentic and anonymous data sharing with forward security," *IEEE Trans. Comput.*, vol. 64, no. 4, pp. 971–983, Apr. 2015.

[11] Y. Gong, Y. Cai, Y. Guo, and Y. Fang, "A privacy-preserving scheme for incentive-based demand response in the smart grid," *IEEE Trans. Smart Grid*, vol. 7, no. 3, pp. 1304–1313, May 2016.

[12] A. Rial and G. Danezis, "Privacy-preserving smart metering," in *Proc. 10th Annu. ACM Workshop Privacy Electron. Soc. (WPES)*, 2011, pp. 49–60.

[13] A. Molina-Markham, G. Danezis, K. Fu, P. Shenoy, and D. Irwin, "Designing privacy-preserving smart meters with low-cost microcontrollers," in *Financial Cryptography and Data Security*. Berlin, Germany: Springer, 2012, pp. 239–253.

[14] A. Alhothaily, C. Hu, A. Alrawais, T. Song, X. Cheng, and D. Chen, "A secure and practical authentication scheme using personal devices," *IEEE Access*, vol. 5, pp. 11677–11687, 2017.

[15] Z. Erkin, J. R. Troncoso-pastoriza, R. L. Lagendijk, and F. Perez-Gonzalez, "Privacy-preserving data aggregation in smart metering systems: An overview," *IEEE Signal Process. Mag.*, vol. 30, no. 2, pp. 75–86, Mar. 2013.

[16] H. Shen, M. Zhang, and J. Shen, "Efficient privacy-preserving cube-data aggregation scheme for smart grids," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 6, pp. 1369–1381, Jun. 2017.

[17] Z. Wang, "An identity-based data aggregation protocol for the smart grid," *IEEE Trans. Ind. Informat.*, vol. 13, no. 5, pp. 2428–2435, Oct. 2017.

[18] X. Chen, F. Zhang, and K. Kim, "A new ID-based group signature scheme from bilinear pairings," *IACR Cryptol. ePrint Arch.*, vol. 2003, p. 116, Aug. 2003.

[19] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in *Advances in Cryptology—CRYPTO*. Berlin, Germany: Springer, 2004, pp. 41–55.

[20] GIGYA, "The 2015 state of consumer privacy & personalization," GIGYA, Palo Alto, CA, USA, White Paper 072015, 2015.

[21] M. Bellare and V. T. Hoang, "Identity-based format-preserving encryption," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.* New York, NY, USA: ACM, Oct. 2017, pp. 1515–1532, doi: 10.1145/3133956.3133995.

[22] C.-H. Tai, P.-J. Tseng, P. S. Yu, and M.-S. Chen, "Identity protection in sequential releases of dynamic networks," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 3, pp. 635–651, Mar. 2014.

[23] B. Reaves, L. Blue, H. Abdullah, L. Vargas, P. Traynor, and T. Shrimpton, "AuthentiCall: Efficient identity and content authentication for phone calls," in *Proc. 26th USENIX Secur. Symp. (USENIX Secur.)* Vancouver, BC, Canada: USENIX Association, 2017, pp. 575–592. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/reaves

[24] H. S. Karimi, K. Jhala, and B. Natarajan, "Impact of real-time pricing attack on demand dynamics in smart distribution systems," in *Proc. North Amer. Power Symp. (NAPS)*, Sep. 2018, pp. 1–6.

[25] P. Wood, S. Bagchi, and A. Hussain, "Profiting from attacks on real-time price communications in smart grids," in *Proc. 9th Int. Conf. Commun. Syst. Netw. (COMSNETS)*, Jan. 2017, pp. 158–165.

[26] P. Wood, S. Bagchi, and A. Hussain, "Defending against strategic adversaries in dynamic pricing markets for smart grids," in *Proc. 8th Int. Conf. Commun. Syst. Netw. (COMSNETS)*, Jan. 2016, pp. 1–8.

[27] X. Yang, X. Zhang, J. Lin, W. Yu, X. Fu, and W. Zhao, "Data integrity attacks against the distributed real-time pricing in the smart grid," in *Proc. IEEE 35th Int. Perform. Comput. Commun. Conf. (IPCCC)*, Dec. 2016, pp. 1–8.

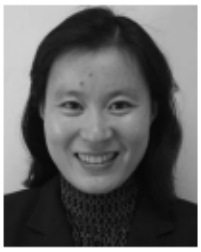
[28] X. Zhang, X. Yang, J. Lin, G. Xu, and W. Yu, "On data integrity attacks against real-time pricing in energy-based cyber-physical systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 1, pp. 170–187, Jan. 2017.

[29] Z. Sui, M. Niedermeier, and H. de Meer, "TAI: A threshold-based anonymous identification scheme for demand-response in smart grids," *IEEE Trans. Smart Grid*, vol. 9, no. 4, pp. 3496–3506, Jul. 2018.

[30] L. Sweeney, "k-anonymity: A model for protecting privacy," *Int. J. Uncertainty, Fuzziness Knowl.-Based Syst.*, vol. 10, no. 5, pp. 557–570, Oct. 2002.

[31] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam, "L-diversity: Privacy beyond k-anonymity," in *Proc. 22nd Int. Conf. Data Eng. (ICDE)*, 2006, p. 24.
[32] N. Li, T. Li, and S. Venkatasubramanian, "T-closeness: Privacy beyond k-anonymity and l-diversity," in *Proc. IEEE 23rd Int. Conf. Data Eng.*, Apr. 2007, pp. 106–115.
[33] K. Rajendran, M. Jayabalan, and M. Ehsan, "A study on k-anonymity, l-diversity, and t-closeness techniques focusing medical data," *Int. J. Comput. Sci. Netw. Secur.*, vol. 17, p. 172, Dec. 2017.
[34] C. Dwork, "Differential privacy: A survey of results," in *Proc. Int. Conf. Theory Appl. Models Comput.* Berlin, Germany: Springer, 2008, pp. 1–19.
[35] C. Hu, J. Yu, X. Cheng, Z. Tian, K. Akkaya, and L. Sun, "CP_ABSC: An attribute-based signcryption scheme to secure multicast communications in smart grids," *Math. Found. Comput.*, vol. 1, no. 1, pp. 77–100, 2018.
[36] C. Hu, X. Cheng, Z. Tian, J. Yu, K. Akkaya, and L. Sun, "An attribute-based signcryption scheme to secure attribute-defined multicast communications," in *Proc. SecureComm*, Dallas, TX, USA, Oct. 2015, pp. 418–437.
[37] Z. Qin, T. Yu, Y. Yang, I. Khalil, X. Xiao, and K. Ren, "Generating synthetic decentralized social graphs with local differential privacy," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.* New York, NY, USA: ACM, Oct. 2017, pp. 425–438, doi: 10.1145/3133956.3134086.
[38] B. Mei, Y. Xiao, R. Li, H. Li, X. Cheng, and Y. Sun, "Image and attribute based convolutional neural network inference attacks in social networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 2, pp. 869–879, Jun. 2020.
[39] Y. Xiao and L. Xiong, "Protecting locations with differential privacy under temporal correlations," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.* New York, NY, USA: ACM, Oct. 2015, pp. 1298–1309, doi: 10.1145/2810103.2813640.
[40] S. Wang, X. Meng, J. Yu, R. Bie, Y. Sun, and X. Cheng, "*N*-in-one: A novel location-based service," *IEEE Trans. Veh. Technol.*, vol. 67, no. 6, pp. 5274–5286, Jun. 2018.
[41] M. Abu Alsheikh, D. Niyato, D. Leong, P. Wang, and Z. Han, "Privacy management and optimal pricing in people-centric sensing," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 4, pp. 906–920, Apr. 2017.
[42] L. Zhu, X. Tang, M. Shen, X. Du, and M. Guizani, "Privacy-preserving DDoS attack detection using cross-domain traffic in software defined networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 628–643, Mar. 2018.
[43] C. Hu, F. Zhang, X. Cheng, X. Liao, and D. Chen, "Securing the communications to wireless body area networks," in *Proc. 2nd ACM Workshop Hot Topics Wireless Netw. Secur. Privacy (HotWiSec)*, Budapest, Hungary, Apr. 2013, pp. 31–36.
[44] F. K. Dankar and K. E. Emam, "Practicing differential privacy in health care: A review," *Trans. Data Privacy*, vol. 6, no. 1, pp. 35–67, Apr. 2013. [Online]. Available: http://dl.acm.org/citation.cfm?id=2612156.2612159
[45] C. Hu, X. Cheng, F. Zhang, D. Wu, X. Liao, and D. Chen, "OPFKA: Secure and efficient ordered-physiological-feature-based key agreement for wireless body area networks," in *Proc. IEEE INFOCOM*, Turin, Italy, Apr. 2013, pp. 2274–2282.
[46] S. Ghayyur *et al.*, "IoT-detective: Analyzing IoT data under differential privacy," in *Proc. Int. Conf. Manage. Data* New York, NY, USA: ACM, May 2018, pp. 1725–1728, doi: 10.1145/3183713.3193571.
[47] 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 Things J.*, vol. 4, no. 6, pp. 1844–1852, Dec. 2017.
[48] A. Bhardwaj and S. Som, "Study of different cryptographic technique and challenges in future," in *Proc. Int. Conf. Innov. Challenges Cyber Secur. (ICICCS-INBUSH)*, Feb. 2016, pp. 208–212.
[49] S. Izhar, A. Kaushal, R. Fatima, and M. A. Qadeer, "Enhancement in data security using cryptography and compression," in *Proc. 7th Int. Conf. Commun. Syst. Netw. Technol. (CSNT)*, Nov. 2017, pp. 212–215.
[50] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A survey on homomorphic encryption schemes: Theory and implementation," *ACM Comput. Surv.*, vol. 51, no. 4, pp. 79-1–79-35, Jul. 2018, doi: 10.1145/3214303.
[51] A. Beimel, "Secret-sharing schemes: A survey," in *Proc. 3rd Int. Conf. Coding Cryptol. (IWCC)*. Berlin, Germany: Springer-Verlag, 2011, pp. 11–46. [Online]. Available: http://dl.acm.org/citation.cfm?id=2017916.2017918
[52] D. Boneh, A. Sahai, and B. Waters, "Functional encryption: A new vision for public-key cryptography," *Commun. ACM*, vol. 55, no. 11, pp. 56–64, Nov. 2012, doi: 10.1145/2366316.2366333.
[53] M. Wen, X. Zhang, H. Li, and J. Li, "A data aggregation scheme with fine-grained access control for the smart grid," in *Proc. IEEE 86th Veh. Technol. Conf. (VTC-Fall)*, Sep. 2017, pp. 1–5.
[54] G. Zhuo, Q. Jia, L. Guo, M. Li, and P. Li, "Privacy-preserving verifiable data aggregation and analysis for cloud-assisted mobile crowdsourcing," in *Proc. IEEE 35th Annu. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr. 2016, pp. 1–9.
[55] C. Huang, D. Liu, J. Ni, R. Lu, and X. Shen, "Reliable and privacy-preserving selective data aggregation for fog-based IoT," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.
[56] R. Li, C. Sturtivant, J. Yu, and X. Cheng, "A novel secure and efficient data aggregation scheme for IoT," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1551–1560, Apr. 2019.
[57] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," in *Advances in Cryptology—CRYPTO*. Berlin, Germany: Springer, 2001, pp. 213–229.
[58] J. C. Choon and J. H. Cheon, "An identity-based signature from gap Diffie-Hellman groups," in *Public Key Cryptography—PKC*. Berlin, Germany: Springer, 2002, pp. 18–30.
[59] D. Chaum and E. Van Heyst, "Group signatures," in *EUROCRYPT*. Berlin, Germany: Springer, 1991, pp. 257–265.
[60] M. Rabin, "Digital signatures and public key functions as intractable as factoring," Lab. Comput. Sci., MIT, Cambridge, MA, USA, Tech. Memo TM-212, 1979.
[61] J. Liu, Y. Xiao, and J. Gao, "Achieving accountability in smart grid," *IEEE Syst. J.*, vol. 8, no. 2, pp. 493–508, Jun. 2014.
[62] R. Jagadeesan, A. Jeffrey, C. Pitcher, and J. Riely, "Towards a theory of accountability and audit," in *Computer Security—ESORICS*. Berlin, Germany: Springer, 2009, pp. 152–167.
[63] R. Küsters, T. Truderung, and A. Vogt, "Accountability: Definition and relationship to verifiability," in *Proc. 17th ACM Conf. Comput. Commun. Secur. (CCS)*, 2010, pp. 526–535.
[64] J. Feigenbaum, A. D. Jaggard, and R. N. Wright, "Towards a formal model of accountability," in *Proc. Workshop New Secur. Paradigms Workshop (NSPW)*, 2011, pp. 45–56.
[65] C. Ko *et al.*, "Analysis of an algorithm for distributed recognition and accountability," in *Proc. 1st ACM Conf. Comput. Commun. Secur. (CCS)*, 1993, pp. 154–164.
[66] W. Dai. (2009). *Crypto++ 5.6. 0 Benchmarks*. [Online]. Available: http://www.cryptopp.com/benchmarks.html
[67] D. Freeman, M. Scott, and E. Teske, "A taxonomy of pairing-friendly elliptic curves," *J. Cryptol.*, vol. 23, no. 2, pp. 224–280, Apr. 2010.
[68] I. F. Blake, G. Seroussi, and N. P. Smart, "Pairing," in *Advances in Elliptic Curve Cryptography*, vol. 317. Cambridge, U.K.: Cambridge Univ. Press, 2005.

Chunqiang Hu (Member, IEEE) received the B.S. degree in computer science and technology from Southwest University, Chongqing, China, in 2006, the M.S. and Ph.D. degrees in computer science and technology from Chongqing University, Chongqing, China, in 2009 and 2013, respectively, and the second Ph.D. degree in computer science from The George Washington University, Washington, DC, USA, in 2016. He was a Visiting Scholar with The George Washington University from 2011 to 2012. He is currently a Faculty Member with the School of Big Data & Software Engineering, Chongqing University. His research interests include privacy-aware computing, blockchain, IoT security and privacy, and applied cryptography. He was honored with the Hundred-Talent Program by Chongqing University. He is a member of the ACM.



Xiuzhen Cheng (Fellow, IEEE) received the M.S. and Ph.D. degrees in computer science from the University of Minnesota–Twin Cities, in 2000 and 2002, respectively. She is currently a Professor with the Department of Computer Science, The George Washington University, Washington, DC, USA. Her current research interests include privacy-aware computing, wireless and mobile security, cyber physical systems, mobile computing, and algorithm design and analysis. She has served on the editorial boards of several technical journals and the technical program committees of various professional conferences/workshops. She also has chaired several international conferences. She worked as a Program Director for the U.S. National Science Foundation (NSF) from April to October in 2006 (full time), and from April 2008 to May 2010 (part time). She received the NSF CAREER Award in 2004. She is a member of the ACM.

Zhi Tian (Fellow, IEEE) has been a Professor with the Electrical and Computer Engineering Department, George Mason University, Fairfax, VA, USA, since January 2015. Prior to that, she was on the Faculty of the Michigan Technological University, from 2000 to 2014. Her research interests lie in statistical signal processing and data analytics, wireless communications, and wireless sensor networks. She has served as an Associate Editor for the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS (2002–2008) and the IEEE TRANSACTIONS ON SIGNAL PROCESSING (2006–2009). She is a Distinguished Lecturer of the IEEE Vehicular Technology Society (2013–2015) and the IEEE Communications Society (2015–2016).

Jiguo Yu (Senior Member, IEEE) received the Ph.D. degree from the School of mathematics, Shandong University, in 2004. Since 2007, he has been a Professor with the School of Computer Science, Qufu Normal University, Shandong, China. He is currently a Professor with the School of Information Science and Engineering, Qufu Normal University. His main research interests include wireless networks, distributed algorithms, peer-to-peer computing, and graph theory. In particular, he is interested in designing and analyzing algorithms for many computationally hard problems in networks. He is a Senior Member of the China Computer Federation (CCF).

Weifeng Lv is currently a Professor with the School of Computer Science and Engineering, Beihang University, Beijing, China. He is also with the State Key Laboratory of Software Development Environment, Beihang University, and also with the Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University. His research interests lie in data processing and algorithm design and analysis.