Blockchain Enables Your Bill Safer

Qin Wang, *Student Member, IEEE,* Longxia Huang,
Shiping Chen, *Senior Member, IEEE,* and Yang Xiang, *Fellow, IEEE*

Abstract—As one of the most frequently used IoT devices, energy smart meter has been widely adopted to facilitate the measures of residential energy use. Residents pay for the bills from energy suppliers according to their monthly/seasonal usage. Practically, there is a demand from residents/governments to check whether the bills are in line with their real consumptions. However, it is challenging to realize this demand due to two critical problems. The first problem refers to the non-repudiated privacy issue caused by the access to residents' energy consumption history (*e.g.*, data integrity may be questioned, and residents' daily timetables may be exposed). The second problem comes from the efficiency requirement for bulk auditing requests on residents' bills and consumptions, usually risen by governments. So far, we have not found any solutions that can be directly used in this case.
In this paper, we propose using homomorphic encryption cooperated with blockchain technique to leverage the data auditing and privacy-preserving requirements. We also employe certificateless signature to resolve the efficiency bottleneck in batch auditing. This framework, called *pAuditChain*, not only accepts personal requests from residents for consumption checking, but also handles bulk auditing requests issued by governments. To validate the correctness of the framework functions, we carried out a series of theoretical analysis, especially on the privacy-preserving and auditing processes. To the best of our knowledge, the proposed framework is among the first solutions to improve the security and privacy of bills without losing the auditing function. Our approach concerns with IoT smart meters in energy supply industries, and could be further extended to other forms of IoT devices with the bill demands.

Index Terms—IoT (meter), Blockchain, Auditing, Privacy

1 Introduction

A smart meter (also known as an advanced meter or 'type 4' meter) is a frequently used IoT device that digitally measures residents' energy use. which measures when and how much electricity/gas is used at residents' premises. It sends their consumption information back to energy retailers remotely, without the meters being manually read by a meter reader. Smart meters can also do other things like allowing the energy supply to be switched on/off without a field technician, measuring the power quality at residents' premises, and notifying their energy suppliers when the electricity/gas goes out, which greatly benefits users with convince.

Every month or season, energy retailers will generate and send bills to residents according to their consumptions reported by smart meters. However, there is always a demand from 1) residents to check the correctness of bills and 2) governments to audit residents' energy consumption data. This demand is basically rooted from long-standing distrust of people on digital devices' security issues. Although security and data privacy have long been an important consideration for the deployments, state-of-art security technologies are still far from being enough to protect the devices as well as their communication [1].

Taking Australia for example, two types of smart meters have been deployed by Australian energy suppliers. The first is based on the technology proposed by Silver Springs Networks (SSN) [2], and the second is based on technology proposed by GridNet [3]. To date, neither SSN nor GridNet has reported any major breaches of security. However, as declared by the Victoria state government in Australia [4], both are actually subject to attempts by hackers to compromise their devices. This report further suggested that problems may be emerging in enabling the AMI communication systems for certain suppliers. The same problems also exist in other countries [5]. Therefore, it has become a long-standing and reasonable demand for data auditing in energy supply industries from both residents and governments.

It is very challenging to realise the above demand in the real world. First, the access to energy consumption history will lead to critical problem of residents' private information. For example, by applying simple statistics on one's hourly consumption of electricity, hackers can easily disclose the time when the victim resident is at home or not. Hackers are able to commit other crimes, given the information that no one is inside the victim's house. Even if residents' private information can be preserved, as the data is open for access by residents, retailers, or governments, the data integrity may be destroyed by any side during the auditing processes. Second, the governments may periodically rise bulk data auditing requests on residents' consumption history. Considering the large amount of data generated by smart meters, it is very likely to have an efficiency bottleneck for data auditing. These problems will consequently hinder the establishment from energy supply industries.

According to our literature review, there are several similar works under the data auditing umbrella [6], [7], [8], [9], [10], [11]. Although these works have already taken privacy preserving issues as an important consideration in the context of their methods, their designs did not provide non-repudiation functions to the consumption data. As explained in the last paragraph, any side (*e.g.*, resi-

Qin Wang, and Yang Xiang are with the School of Software and Electrical Engineering, Swinburne University of Technology, Melbourne, Australia; Email: {qinwang, yxiang}@swin.edu.au.
Longxia Huang is with the School of Computer Science & Communication Engineering, Jiangsu University, Jiangsu, China. Email: hlongxia2017@163.com.
Shiping Chen is with CSRIO, Data61, Sydney, Australia. Email: shiping.Chen@data61.csiro.au.

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.2020.3016721, IEEE Internet of Things Journal

dents, retailers, governments) could deny the integrity of history data retrieved from databases in previous privacy preserving schemes. In addition, since all the works are built on top of Public Key Infrastructure (PKI), they confronted the disadvantages like complex certificate management and key escrow problem. These drawbacks also disabled the direct deployment of previous methods onto the energy consumption data auditing scenario, particularly in the batch auditing which usually needs very high requirement on efficiency. So far, we have not found any capable solutions in the literature of related fields, which can exactly realise the above demand, particularly for energy supply industries.

In this paper, we are motivated to develop a data auditing framework with a series of innovative schemes that can address the problems or drawbacks in previous works. The framework is illustrated in Fig. 1. We summarised the contributions as follows:

- We introduce homomorphic encryption to leverage the data auditing and privacy preserving requirements. The scheme is also equipped with blockchain technique as a data ledger to avoid repudiation problem when residents' private data could be accessed for auditing. The blockchain(s) is/are designed to be deployed in cloud by governments.
- We employ certificateless signature to resolve the efficiency bottleneck in bulk data auditing. The proposed framework, namely *pAuditChain*, not only accepts personal request from residents for bill checking, but also handles bulk data auditing requests generally risen from governments.
- All the schemes have been theoretically proven or analysed to demonstrate the correctness and robustness of the proposed framework. We believe the idea of *pAuditChain* can also be borrowed by other isomorphic data auditing scenarios in industrial Internet-of-Things (IoT).

The rest of the paper is structured as follows. Section 2 introduces cryptographic building blocks. Section 3 provided a brief design of our system model. Section 4 then detailedly descirbe the pAduditChain framework, followed by Section 5 presenting the comprehensive evaluation of serials analyses. Section 6 summarise the related works referred in our framework. At last we conclude the content of this paper in Section 7.

2 Preliminaries

This section introduces the basic knowledge used in this paper as follows.

2.1 Certificateless signatures

Certificateless signatures (CLS) [12] is used to manage certificates. CLS mitigates the key escrow problem in Identity-based Cryptosystems. Key Generation Center (KGC) generates a share of (partial) private key of users, and the rest share of private key is created by the user himself. Thus, it weakens the power of KGC, who cannot maliciously create a valid signature by his own.

- **Setup**: It inputs a security parameter, and outputs the master key for KGC and parameters for our system.

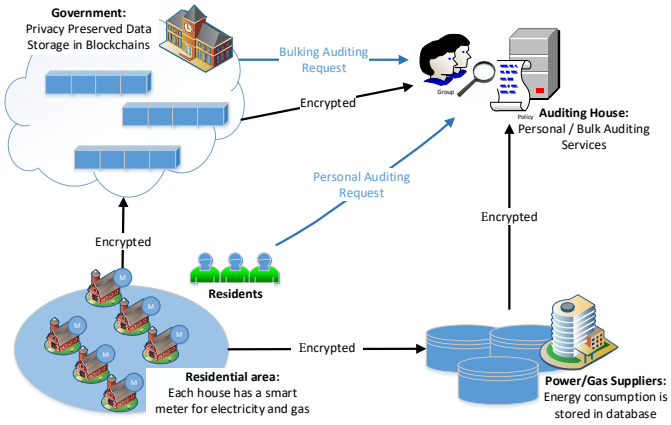


Figure 1. System framework illustration: 1) Residential area is the source of energy consumption data; 2) the data will be transmitted to energy suppliers for generating bills; 3) meanwhile, the data will also be transmitted to government sponsored back-up centre based on blockchain and cloud technologies; 4) residents/government can request personal/bulk audit while access to residents' data during the auditing will not break residents' privacy.

- **Partial-Private-Extract**: A user generate his own share of private key, and register the private key and his identity to KGC.
- **KeyGen**: KGC generate an ID-based private key for registered users.
- **Sign**: Holding the key pair and the public parameters, a user is allowed to sign a message.
- **Verify**: This algorithm checks and verifies the correctness of the signature.

2.2 Paillier Cryptosystem

The privacy is based on the Paillier cryptosystem to hide the amounts of transaction bills as in [13]. The Paillier cryptosystem is an additional homomorphic encryption scheme with the property $\text{Dec}_{sk}(\text{Enc}_{pk}(m_1) \cdot \text{Enc}_{pk}(m_2)) = (m_1 + m_2)$.

- **KeyGen**: Suppose that $n = p \cdot q$, then compute $\lambda(n) = lcm(p-1, q-1)$ and randomly choose $g \in \mathbb{G}$ satisfying $\gcd(L(g^\lambda \bmod n^2), n) = 1$. $\mathbb{G}$ satisfies $\mathbb{G} = \{w | w \in \mathbb{Z}^*_{n^2}\}$. (n, g) is the public key while (p, q) is the private key.
- **Encryption**: Take the plaintext m and a random value as the inputs, output the ciphertext $c = g^m r^n \bmod n^2$.
- **Decryption**: Input the encrypted message c, and the output $m = \frac{L(c^\lambda \bmod n^2)}{L(g^\lambda \bmod n^2)} \bmod n$.

3 Methodology

This section introduces our system model in both personal data auditing and bulk data auditing. The scheme protects the data privacy while being correctly audited.

3.1 System model for personal end

At the personal end, pAuditChain employs Paillier cryptosystem to hide the sensitive amounts in transaction bills. The architecture of the personal end is shown in Fig.2.

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.2020.3016721, IEEE Internet of Things Journal

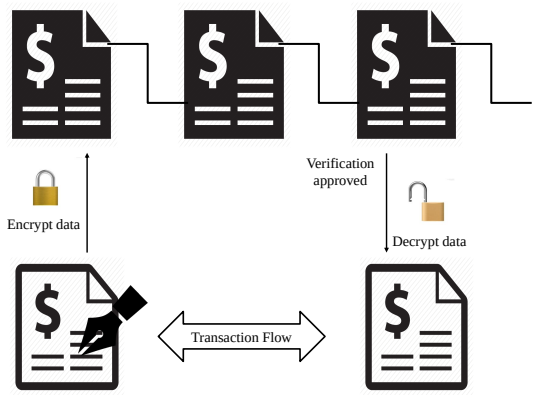


Figure 2. Scheme model for personal auditing end. The individual user in our scheme can achieve the strong anonymity. The homomorphic encryption seal the plain data shown on script into unreadable cipher text while the encrypted data can outcomes correct values under the homomorphic operations. Whenever the verification turns out true, the receiver decrypts the ciphertext to obtain the value he/she needs.

Three entities are included, namely sender, receiver and blockchain ledger. The sender sends encrypted messages to blockchain, from which the receiver obtains the messages.

Specifically, the sender encrypts the bills at the local client, and sends the encrypted messages to the ledger. Encrypted bills are operated and executed on the blockchain. Since the bills hidden in ciphertexts cannot be directly calculated as they are in plaintexts, we employ the additional homophobic encryption cryptosystem for the calculations. The cryptosystem ensures that the calculated ciphertext could be correctly decrypted. Finally, the receivers obtains the encrypted bills and decrypt them.

3.2 Scheme model for bulk end

The architecture of the certificateless public verification system in our bulk auditing scheme is shown in Fig. 3. It includes four entities, including user group, database, KGC, and public verifier.

Specifically, the user group means users who generate the data files such as bills. Each data owner is required to use the private key to sign files before uploading. The private key is generated by the user and KGC. The user needs to registered his/her ID before making KGC to generate a partial private key. The correctness and integrity of outsourced data is ensured by a public (honest) verifier.

The verification is realised by a challenge-and-response protocol. Firstly, the bill owner sends a request to the public verifier for integrity checking. Once receiving the request, the verifier creates a challenge messages and sends them to the database. Then, the database replies by sending a proof message to the verifier. Last, the verifier confirms its correctness and returns the result to the data owner.

4 pAuditChain framework

4.1 Problem Statement

The data generated from IoT devices is always ignorant by people. Anyone with evil purpose can use the collected data to image a clear figure of person. Whenever our behaviours are precisely analysed by enterprises or attackers, they can sale our confidentially personal information for profits. As a result, our life might be disrupted by unpleasant advertisements. The privacy of blockchain covers many fields and we focus on the sensitive values shown on its script. The value can expose many kind of information such as financial value, used energy, recorded numbers and so on. Traditionally, blockchain is anonymous thanks to its unlinkable addresses with real entities. However, a potential risk is the explosion of the values in transaction. The plain values face the risk of the sensitive information leakage. The advertisers can use the absolute values of amounts to precisely analyse a person as we mentioned before. To improve the privacy of individual information from IoT devices, we attempt to hide the values through the homomorphic cryptosystem which ensures the values can be correctly calculated. Our scheme proposes a method to realize confidential transaction to prevent information leakage. Meanwhile, it is compatible with verification schemes to confirm the equality of inputs and outputs.

4.2 pAuditChain privacy scheme for personal end

Homomorphic encryption is the essential building block in our proposed system. The crypto scheme is build on top of underlying blockchain systems by replacing the original plaintext of bills with encrypted ones in their *amount* fields. Thus, the transparent values on bills are hidden without exposure to the governments or public verifies. The encrypted values are sent to the blockchain for further operations/calculations. The key of correct operations is to guarantee its inputs-sum equals to the output-sum. Additionally, another requirement is to ensure the carried values are positive, in case of the negative ones (which means the stolen of electricity). Therefore, we concludes five algorithms towards the personal end of our privacy scheme: **Setup**, **KeyGen**, **Encrypt**, **Verify**, and **Decrypt**.

Setup$(1^\lambda) \rightarrow (pms)$: The algorithm inputs a security parameter 1^λ and it outputs system parameters pms. The algorithm generates parameters (g_α, h_α) and (g_β, h_β) satisfying that $g_\beta = g_d$ and $n_\beta = n_i^{\star 2}$, where g_α is the generator in $\mathbb{Z}^*_{n_\alpha}$ and h_α is the element of the group generated by g_α, and the same with (g_β, h_β).

KeyGen$(pms) \rightarrow (pk_i, pk_i^\star)$: The algorithm inputs system parameters pms and outputs the key pair $(pk_i, pk_i^\star)$. The system generates the prime parameters (p_i, q_i), the private key $sk_i = \lambda_i$ where $\lambda_i = \text{lcm}(p_i - 1)(q_i - 1)$ and the public key $pk_i = (n_i, g_i)$, where $n_i = p_i q_i$ and $g_i \in \mathbb{Z}_{n_i^2}$. Meanwhile, the system generates the public key $pk_i^\star = (n_i^\star, g_i^\star)$ for the verification, where $n_i^\star = p_i^\star q_i^\star$ and $g_i^\star \in \mathbb{Z}_{n_i^2}$.

Encrypt$(m_i, pk_i, pk_i^\star) \rightarrow (c_i, c_i^\star)$: The algorithm inputs the plain amounts m_i under the user pk_i and outputs the ciphertexts c_i. The sender encrypts the bill m_i both under the receiver's pk_i as $c_i = \text{Enc}_{pk_i}(m_i) = g_i^{m_i} r_i^{n_i} \bmod n_i^2$ and system's $pk_i^\star$ as $c_i^\star = \text{Enc}_{pk_i^\star}(m_i) = (g_i^\star)^{m_i} (r_i^\star)^{n_i^\star} \bmod n_\beta$. The system encryption, in the form of commitment number, is used to confirm the equality between inputs and outputs. Note that, the random value $r_i^\star = h_\beta$. h_β is an element in the group which is generated by $g_\beta \in \mathbb{Z}^*_{n_\beta}$

Verify$(V_\alpha, V_\beta, \prod c_i^\star, m, m_i) \rightarrow True/False$: The sender sends his bills c_i through the transaction under pk_i and the bill $c_i^\star$ under $pk_i^\star$. The algorithm is used to verify the output whether equals to input as follows.

The sender generates two committed number $E = E_\alpha(m, r_\alpha) = g_\alpha^m h_\alpha^{r_\alpha}$, $F = E_\beta(\sum m_i, r_\beta) = g_\beta^m h_\beta^{r_\beta}$, where $r_\alpha \in \{-2^s n + 1, ..., 2^s n - 1\}$ and $r_\beta = n_d \in \{-2^s n + 1, ..., 2^s n - 1\}$ in which s is security parameter. The verification of the equality between the input and output needs to prove that: *Step1*: The private $m = \sum m_i$ in E and F from the sender equal; *Step2*: The cipher $H = \prod c_i^\star$ equals to committed value F.

To verify the *Step1*, the protocol follows

1) The sender picks random $\omega \in \{1, ..., 2^{i+t}b - 1\}, \eta_\alpha \in \{1, ..., 2^{l+t+s}n - 1\}, \eta_\beta \in \{1, ..., 2^{l+t+s}n - 1\}$. Then compute $W_\alpha = g_\alpha^\omega h_\alpha^{\eta_\alpha} \bmod n_\alpha$, $W_\beta = g_\beta^\omega h_\beta^{\eta_\beta} \bmod n_\beta$.
2) The sender computes $u = H(W_\alpha || W_\beta)$.
3) The sender computes $D = \omega + um$, $D_\alpha = \eta_\alpha + ur_\alpha$, $D_\beta = \eta_\beta + ur_\beta$ and sends $(u, D, D_\alpha, D_\beta)$ to the verification layer.
4) The verification layer checks whether $u = u'$ where $u' = H(g_\alpha^D h_\alpha^{D_\alpha} E^{-u} \bmod n_\alpha || g_\beta^D h_\beta^{D_\beta} F^{-u} \bmod n_\beta)$.

If the *Step1* successes, then we verify the *Step2*, and the protocol follows

1) The verification layer computes the ciphers it received $H = \prod c_i^\star = c_1' c_2' ... c_i' = (g_i^\star)^{\sum m_i'} (r_i^\star)^{n_i^\star} \bmod (n_i^\star)^2$.
2) From above, we can randomly select $r_i^\star = h_\beta$ in encryption and $r_\beta = n_i^\star$ in verification. From KeyGen, we obtain $n_\beta = (n_i^\star)^2$, and $g_i^\star = g_\beta$. Thus, we have $H = (g_i^\star)^{\sum m_i'} (r_i^\star)^{n_i^\star} \bmod (n_i^\star)^2$, $F = g_\beta^{\sum m_i} h_\beta^{r_\beta} \bmod n_\beta$.
3) Verify whether $H == F$, if no, abandon the transaction, and if yes, in True.

Decrypt $(sk_i, c_i) \rightarrow (m_i)$: The algorithm inputs the encrypted amounts c_i and private key sk_i and outputs the decrypted bill m_i. The receiver obtains the encrypted ciphers c_is and use their sk_i to decrypt them: $m_i = \frac{L(c_i^{\lambda_i} \bmod n_i^2)}{L(g_i^{\lambda_i} \bmod n_i^2)} \bmod n_i$ where $L(x) = \frac{x-1}{n}$ and $x \in \mathcal{S}_n = \{u < n^2 | x = 1 \bmod n\}$.

4.3 pAuditChain on bulk end

This section introduces the details of our proposed signature scheme and certificateless public auditing scheme.

4.3.1 Signature Scheme

Our signature scheme includes five parts.

Setup$(1^k) \rightarrow (x, \mathbb{G}_1, \mathbb{G}_2, q, e, P, T, H_1, H_2, P_{pub})$: It inputs a security parameter 1^k, and outputs a master key x and system parameters $pms = (\mathbb{G}_1, \mathbb{G}_2, q, e, P, T, H_1, H_2, P_{pub})$. Suppose that the group $\mathbb{G}_1$ is an additive cyclic group, $\mathbb{G}_2$ is the multiplicative group, and $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is a bilinear map, P, T are two generators of $\mathbb{G}_1$ and $\mathbb{G}_2$. H_1 and H_2 are hash functions satisfying $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$ and $H_2 : \mathbb{G}_1 \times \{0,1\}^* \rightarrow \mathbb{Z}_q$, once getting the security parameter 1^k, KGC randomly selects a parameter $x \in \mathbb{Z}_q$ to compute the master public key as $P_{pub} = x \cdot P$. Last, KGC stores its master private key x and send the parameters $pms = (\mathbb{G}_1, \mathbb{G}_2, q, e, P, T, H_1, H_2, P_{pub})$ to users.

Partial-Private-Extract$(ID_i) \rightarrow (x_{ID_i}, P_{ID_i})$: The algorithm is run by a user to generate his partial private key. The user u_i (equally ID_i) randomly creates x_{ID_i} as his partial private key and computes the corresponding partial public key P_{ID_i} as $x_{ID_i} \cdot P$. Then, the algorithm registered the identity to KGC to obtain the rest shares of his private key.

KeyGen$(ID_i) \rightarrow (s_{ID_i}, R_{ID_i})$: The algorithm is run by KGC to compute the private key for users with their identity ID_i. It inputs a user's ID_i, user's public key P_{ID}, master private key x, and system parameters pms. Then the algorithm ouptuts a private key associated with ID. Specifically, KGC selects a random parameter r_{ID_i} and computes two auxiliary parameters $R_{ID_i} = r_{ID_i} \cdot P$ and $h_{ID_i} = H_1(ID_i, R_{ID_i}, P_{ID_i})$.

Then KGC computes $s_{ID_i} = r_{ID_i} + h_{ID_i} \cdot x$ and sends the partial key pair (s_{ID_i}, R_{ID_i}) to the user. Upon obtaining (s_{ID_i}, R_{ID_i}), the user checks the validity of s_{ID_i} by verifying whether the equality of the following equation holds:

$$s_{ID_i} \cdot P = R_{ID_i} + h_{ID_i} \cdot P_{pub}$$

If and only if it holds, the user will accept the partial key pair. And then, if the partial key pair is correct, we have $s_{ID_i} = r_{ID_i} + h_{ID_i} \cdot x$, $R_{ID_i} = r_{ID_i} \cdot P$ and $x_{ID_i} \cdot P = P_{pub}$. Therefore, we have

$$\begin{aligned} s_{ID_i} \cdot P &= (r_{ID_i} + h_{ID_i} \cdot x)P \\ &= r_{ID_i} P + h_{ID_i} \cdot x \cdot P \\ &= R_{ID_i} + h_{ID_i} \cdot P_{pub}. \end{aligned}$$

Last, the private key of the user is formed as $sk_{ID} = (x_{ID}, s_{ID})$ and the public key is $pk_{ID} = (P_{ID}, R_{ID})$.

Sign$(ID_i, pms) \rightarrow (s, m)$: The algorithm generates a signature for the file block. A user with identity ID_i inputs the parameters $pms = (\mathbb{G}_1, \mathbb{G}_2, q, e, P, T, H_1, H_2, P_{pub})$, the private key $sk_{ID_i} = (s_{ID_i}, x_{ID_i})$, and a file block m, and generates a signature of the file block m. During this process, the user first computes $h = H_2(m || index || ID_i)$ and the signature as $s = h \cdot s_{ID_i} + x_{ID_i} \cdot m \cdot T$. Finally, the user uploads (s, m) to the database server.

Verify$(s, m) \rightarrow (s, m)$: The algorithm verifies the correctness of a signature uploaded by a user. After receiving the signature and the file block pair (s, m), the database sever is required to verify the correctness of the signature before saving. The database server first computes $h_{ID_i} = H_1(ID_i, R_{ID_i}, P_{ID_i})$, and then verifies whether $e(s, P) = e(T, m \cdot P_{ID_i}) \cdot e(h, R_{ID_i}) \cdot e(h \cdot h_{ID_i}, P_{pub})$ holds. If the signature passes the verification, through

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.2020.3016721, IEEE Internet of Things Journal

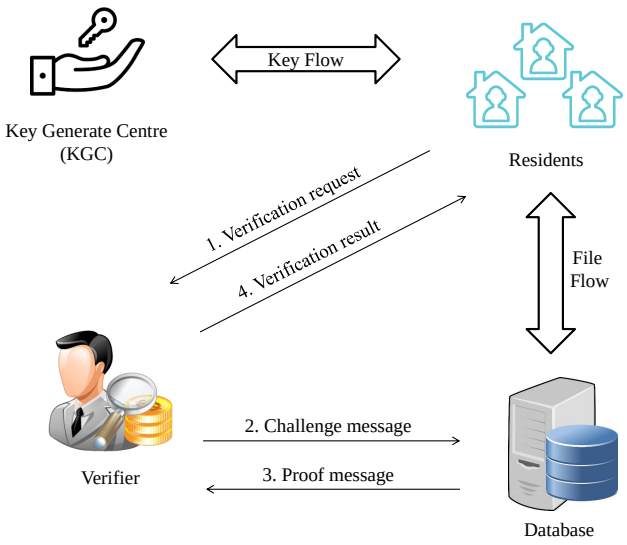


Figure 3. System model on bulk auditing end. The system has four entities including KGC, user group (residents), verifier (government) and database sever (service provider). The different entities communicate with each other through challenge-and-response protocol. The verifier needs to confirm the integrity of files in the database.

$s = x_{ID_i} \cdot m \cdot T + h \cdot s_{ID_i}$ and $s_{ID_i} = r_{ID_i} + h_{ID_i} \cdot x$, we have the equation,

$$\begin{aligned} e(s,P) &= e\left(x_{ID_i} \cdot m \cdot T + h \cdot s_{ID_i}, P\right) \\ &= e\left(x_{ID_i} \cdot m \cdot T, P\right) \cdot e\left(h \cdot (r_{ID_i} + h_{ID_i} \cdot x), P\right) \\ &= e\left(T, m \cdot P_{ID_i}\right) \cdot e\left(h, R_{ID_i}\right) \cdot e\left(h \cdot h_{ID_i}, P_{pub}\right) \end{aligned}$$

4.3.2 Certificateless Public Auditing Scheme

This section provides an efficient certificateless public auditing scheme based on CLS scheme. There are total four entities in this model: key generation centre (KGC), users, public verifier, and database server. We mapped these abstract entities into our specific scenarios, as shown in Fig.3. The user is the resident to send/receive bills, the verifier is mapped to the government who apply the request for regular checking, and the database is maintained and managed by the service providers to offer essential services and bills. Besides, we assume KGC is set up by a trusted third party, as its assumptions in previous cryptosystems. Then, back to the auditing scheme, seven algorithms are included. We highlight the last three algorithms (**Sign**, **ProofGen** and **ProofVerify**), which are different to those in Section 2.1.

Sign$(pms, sk_{ID}, M) \rightarrow (S, M)$: This algorithm inputs the system parameters pms, user's private key $sk_{ID} = (x_{ID}, s_{ID})$, and the message M where M is divided into several parts $M = m_1 || \cdots || m_n$, and outputs a signature. Specifically, for $1 \leq i \leq n$, the user calculates $h_i = H_2(m_i || index || ID)$ and computes the signature $s_i = x_{ID} \cdot m_i \cdot T + h_i \cdot s_{ID}$. The user uploads the signature $S = (s_1, \cdots, s_n)$ and the message $M = (m_1, \cdots, m_n)$ to the database server.

ProofGen$(M, S) \rightarrow (chall, Por)$: It is necessary for a verifier to verify the integrity of files in the database since the bill owner may not store files at local. The verifier randomly selects a c-element subset I and the number ℓ to create a challenge messages $chall = \{\ell, I\}$ and send the messages to the server. Once receiving $chall = \{\ell, I\}$, the sever computes $C = (i, v_i)$ where $i \in I, v_i = \ell^i \mod q$. Based on the file $M = (m_1, \cdots, m_n)$, the server, then, computes $S = \sum_{i \in I} v_i \cdot s_i$, $H = \sum_{i \in I} v_i \cdot h_i$, $\mu = \sum_{i \in I} v_i \cdot m_i$ and sends proof $Por = \{S, H, \mu\}$ to the verifier.

ProofVerify$(M, Por) \rightarrow True/False$: Upon obtaining the proof $Por = \{S, H, \mu\}$ from database server, the verifier computes $h_{ID} = H_1(ID, R_{ID}, P_{ID})$ and then verifies the correctness by checking the equality $e(S, P) = e(T, \mu \cdot P_{ID}) \cdot e(H, R_{ID}) \cdot e(H \cdot h_{ID}, P_{pub})$. The verifier accepts the proof if the equality is passed, otherwise reject it. Since $S = \sum_{i \in I} v_i \cdot s_i$, $H = \sum_{i \in I} v_i \cdot h_i$, $\mu = \sum_{i \in I} v_i \cdot m_i$ and $h_{ID} = H_1(ID, R_{ID}, P_{ID})$, we have,

$$\begin{aligned} e(S,P) &= e\left(\sum_{i \in I} v_i s_i, P\right) \\ &= e\left(\sum_{i \in I} v_i \cdot (x_{ID} \cdot m_i \cdot T + h_i \cdot s_{ID}), P\right) \\ &= e\left(\sum_{i \in I} v_i \cdot (x_{ID} \cdot m_i \cdot T + h_i \cdot (r_{ID} + h_{ID} \cdot x)), P\right) \\ &= e\left(T, \mu \cdot P_{ID}\right) \cdot e\left(H, R_{ID}\right) \cdot e\left(H \cdot h_{ID}, P_{pub}\right) \end{aligned}$$

5 Evaluation

5.1 Correctness Analysis

Theorem 1. The committed values inside the ciphertexts stay in specific range.

Proof 1. From CM1[1], the receiver obtains that

$$E_1 = g^y h^r \mod n, \; E_2 = E_1^{\alpha} h^{r_1} = g^{\alpha y} h^{\alpha r + r_1} \mod n$$
$$E_3 = E_2^{\alpha} h^{r_2} = g^{\alpha^2 y} h^{\alpha^2 r + \alpha r_1 + r_2} \mod n$$

From CM2, the receiver obtains that

$$F = g^{\omega} h^{r_3} \mod n, \; V = g^v / E_3 = g^{\omega} h^{-r\alpha^2 - r_1 \alpha - r_2} \mod n$$

and convinces that

$$\begin{aligned} g^v &= V E_3 = g^{\omega} h^{-r\alpha^2 - r_1\alpha - r_2} g^{\alpha^2 y} h^{\alpha^2 r + \alpha r_1 + r_2} \\ &= g^{\alpha^2 y + \omega} h^0 = g^{\alpha^2 y + \omega} \mod n \end{aligned}$$

Here, the sender has no knowledge of the factorization of n. The receiver has confirmed that $v > 2^{t+l+s+T}$ and convinces from CM3 that $\omega \in [-2^{t+l+s+T}, 2^{t+l+s+T}]$. Therefore, we obtains $\boxed{y > 0}$. Based on that, we can prove $y = m_i - a > 0$, and $m_i > a$, where $a = 0$. The correctness analysis can also be suited to each m_i, where $i \in \{1, 2, ..., i\}$. If the results shows that $y < 0$, we have, $v = \alpha^2 y + \omega \leq \omega \leq 2^{t+l+s+T}$, which is **contradicted** with $v > 2^{t+l+s+T}$ as mentioned before.

1. We abbreviate the commitments as CM for short, and use index $\{1, 2, 3\}$ to distinguish them.

Theorem 2. Two committed values inside the ciphertexts are equal to each other.

Proof 2. From the *Step2* protocols, we can see

$$u = H(W_\alpha || W_\beta) = (g_\alpha^\omega h_\alpha^{\eta_\alpha} \bmod n_\alpha || g_\beta^\omega h_\beta^{\eta_\beta} \bmod n_\beta).$$

Suppose committed values E, F are equal, we obtains

$$\begin{aligned} u' =& H(g_\alpha^D h_\alpha^{D_\alpha} E^{-u} \bmod n_\alpha || g_\beta^D h_\beta^{D_\beta} F^{-u} \bmod n_\beta) \\ =& H(g_\alpha^{um+\omega} h_\alpha^{ur_\alpha+\eta_\alpha} (g_\alpha^m h_\alpha^{r_\alpha})^{-u} \bmod n_\alpha || \\ & g_\beta^{um+\omega} h_\beta^{ur_\beta+\eta_\beta} (g_\beta^m h_\beta^{r_\beta})^{-u} \bmod n_\beta) \\ =& g_\alpha^\omega h_\alpha^{\eta_\alpha} \bmod n_\alpha || g_\beta^\omega h_\beta^{\eta_\beta} \bmod n_\beta = u. \end{aligned}$$

We observe that the executed ciphertext H and committed value F:

$$\begin{aligned} H &= \prod c_i' = (g_i^\star)^{\sum m_i'} (r_i^\star)^{n_i^\star} \bmod (n_i^\star)^2 \\ F &= g_\beta^{\sum m_i} h_\beta^{r_\beta} \bmod n_\beta = (g_i^\star)^{\sum m_i} (r_i^\star)^{n_d} \bmod (n_i^\star)^2 \end{aligned}$$

where $r_i^\star = h_\beta$, $r_\beta = n_i^\star$ are picked at random, and $n_\beta = n_d^2$, and $g_i^\star = g_\beta$ are generated as mentioned before. If $H = F$, namely, $H = (g_i^\star)^{\sum m_i'} (r_i^\star)^{n_i^\star} \bmod (n_i^\star)^2 = (g_i^\star)^{\sum m_i} (r_i^\star)^{n_i^\star} \bmod (n_i^\star)^2 = F$, and $\sum m_i' = \sum m_i = m$, meaning that the sum of input values is equal to that of the output values, denoted as $\boxed{H = F}$. More detailed proofs could be referenced in [14].

Theorem 3. The homomorphically operated values can be correctly decrypted.

Proof 3. Since $1+n \in \mathbb{G}$, so $[\![g]\!]_{1+n} = [\![1+n]\!]_g^{-1} \bmod n$ is revertible which results in that $L(g^\lambda \bmod n^2)$ is revertible modulo n. Thus, we obtain

$$\frac{L(\omega^\lambda \bmod n^2)}{L(\omega^\lambda \bmod n^2)} = \frac{\lambda [\![\omega]\!]_{1+n}}{\lambda [\![g]\!]_{1+n}} = \frac{[\![\omega]\!]_g [\![g]\!]_{1+n}}{[\![g]\!]_{1+n}} = [\![\omega]\!]_g \bmod n$$

where $g \in \mathbb{G}$ and $\omega \in \mathbb{Z}_{n^2}^*$.

5.2 Security Analysis

To guarantee the security of our scheme, we provide the security proofs in this section.

Theorem 4. Our scheme is non-forgeable where the adversary $\mathcal{A}$ (KGC/user) are unable to forge a valid signature with a negligible probability.

Proof 4. We analysis two types of adversaries in a secure CLS scheme [12]:

- $\mathcal{A}_1$, representing a dishonest KGC, is able to access the master private key, but unable to forge users' public key.
- $\mathcal{A}_2$, representing a malicious user, unable to access the master private key, but able to replace users' public keys.

Our proof proceeds as follows: Suppose that $\mathcal{A}_1$ or $\mathcal{A}_2$ can forge a valid signature in the scheme, an algorithm $\mathcal{F}$ exists to solve the discrete logarithm problem (*DLP*). Now, we analyse the situations on $\mathcal{A}_1$ and $\mathcal{A}_2$.

Issue($\mathcal{A}_1$): Assume that $\mathcal{A}_1$ is a (t, q_s, q_h, q_c)-forger who can launch an adaptively chosen message attack. $\mathcal{A}_1$ can construct $\mathcal{F}$ to solve DLP.
$\mathcal{F}$ selects $x \in \mathbb{Z}_q$ at random as his master private key, and computes $P_{pub} = x \cdot P$. The algorithm chooses ID^* as the challenged identity ID. Then, the algorithm sends pms and x to the adversary $\mathcal{A}_1$. $\mathcal{A}_1$ asks to $\mathcal{F}$, and $\mathcal{F}$ answers the queries as,

Create(ID): $\mathcal{A}_1$, holding ID_i, asks $\mathcal{F}$ with $(ID, P_{ID}, R_{ID}, h_{ID}, x_{ID}, s_{ID})$. If the identity ID_i has been already stored in the hash list L_C maintained by $\mathcal{F}$, he sends the results back to $\mathcal{A}_1$. If not, $\mathcal{F}$ selects $a, b \in \mathbb{Z}_q$. If $ID \neq ID_i$, $\mathcal{F}$ selects $c \in \mathbb{Z}_q$, sets $R_{ID} = aP$, $PID = bP$, $hID = H_1(ID, P_{ID}, R_{ID}) = c$, $x_{ID} = b$, $s_{ID} = a + h_{ID}x$. Last, $\mathcal{F}$ responses $(ID, P_{ID}, R_{ID}, x_{ID}, h_{ID}, s_{ID})$ and store the $(ID, P_{ID}, R_{ID}, h_{ID})$ in their list L_{H_1}. If $ID = ID_i$, the algorithm $\mathcal{F}$ coumptes $R_{ID} = a \cdot P, P_{ID} = Q, h_{ID} = H1(ID, P_{ID}, R_{ID}) = b, s_{ID} = a + h_{ID} \cdot x, x_{ID} = \perp$.

H_1-query: $\mathcal{A}_1$, holding ID, queries H_1 for h_{ID}. If ID has already stored in L_{H_1}, $\mathcal{F}$ replies with the stored value. If ID does exist, $\mathcal{F}$ invokes *Create(ID)* to obtain the results $(ID, P_{ID}, R_{ID}, x_{ID}, h_{ID}, s_{ID})$ and outputs h_{ID}.

Partial-Private-Extract(ID): If $ID \neq ID^*$, $\mathcal{F}$ invokes *Create(ID)* to obtains $(ID, P_{ID}, R_{ID}, x_{ID}, h_{ID}, s_{ID})$ and returns s_{ID}. If $ID = ID^*$, the algorithm $\mathcal{F}$ stops.

Key-Gen(ID): $\mathcal{F}$ replies the $pk_{ID} = (P_{ID}, R_{ID})$ if the ID has already stored in L_C. Otherwise, $\mathcal{F}$ invokes the oracle *Create(ID)* to obtain the results $(ID, P_{ID}, R_{ID}, x_{ID}, h_{ID}, s_{ID})$ and outputs pk_{ID}. If $ID = ID^*$, the algorithm $\mathcal{F}$ stops. If $ID \neq ID^*$, $\mathcal{F}$ replies with x_{ID}.

H_2-query: $\mathcal{A}_1$, holding ID, queries H_2 to obtain h_{ID}, $\mathcal{F}$ randomly selects $h \in \mathbb{Z}_q$ and computes $h = H_2(m||index||ID)$, $(m, ID, h, R_{ID}, P_{ID})$ and returns h.

Sign(ID, m): $\mathcal{A}_1$, holding ID, queries on (ID, m). If $ID \neq ID^*$, the algorithm $\mathcal{F}$ executes as the steps before. If $ID = ID^*$, $\mathcal{F}$ computes $s = x_{ID} \cdot m \cdot T + h \cdot s_{ID} = x_{ID} \cdot m \cdot T + h \cdot r_{ID} + h_{ID} \cdot x \cdot h$ and returns the results.

Create oracle fails if the inconsistency occurs with the probability at most q_h/n under $H_1(ID, R_{ID}, P_{ID})$. The successful of our oracle is q_c times with the probability $(1-q_h/n)^{q_c}$ at least $1 - \frac{q_h q_c}{n}$. Similarly, the oracle H_2 fails if the inconsistency occurs with the probability at most q_h/n under $H_2(m||index||ID)$. The successful of oracle is q_h times with the probability $(1 - q_h/n)^{q_h}$ at least $1 - \frac{q_h^2}{n}$. Furthermore, the probability of $ID = ID^*$ is $1/q_c$. Combining them together, the successful probability is at most $\frac{1}{q_c}(1 - \frac{q_h q_c}{n})(1 - \frac{q_h^2}{n})$.

Issue($\mathcal{A}_2$): Suppose $\mathcal{A}_2$ is a (t, q_c, q_s, q_h)-forger. $\mathcal{F}$ picks $x \in \mathbb{Z}_q$, sets $P_{pub} = x \cdot P$, selects ID^*, and gives pms and x to $\mathcal{A}_2$ as does in $\mathcal{A}_1$. Then, $\mathcal{F}$ responses $\mathcal{A}_2$'s queries.

Create(ID) : $\mathcal{A}_2$ asks $\mathcal{F}$ to response the message $(ID, P_{ID}, R_{ID}, x_{ID}, h_{ID}, s_{ID})$ with identity ID_i. If ID_i is stored in L_C maintained by $\mathcal{F}$, he sends results to $\mathcal{A}_2$. If not, $\mathcal{F}$ selects $a, b, c \in \mathbb{Z}_q$, computes $R_{ID} = a \cdot P_{pub} + b \cdot P$, $P_{ID}) = a$, $P_{ID} = c \cdot P$, $x_{ID} = c$, $s_{ID} = b$, $h_{ID} = H_1(ID, R_{ID}$.

Last, $\mathcal{F}$ responses $(ID, P_{ID}, R_{ID}, x_{ID}, h_{ID}, s_{ID}$ and inserts $(ID, P_{ID}, R_{ID}, h_{ID})$ into L_{H_1}. In this step, (h_{ID}, R_{ID}, s_{ID}) satisfied the equation $s_{ID} \cdot P = R_{ID} + h_{ID} \cdot P_{pub}$. Equally, the private key is valid.

H_1-query: $\mathcal{A}_2$, holding ID, queries H_1 to obtain h_{ID}, $\mathcal{F}$ responses the stored value if the ID has already stored in L_{H_1}. If not, $\mathcal{F}$ invokes *Create(ID)* to obtain $(ID, P_{ID}, R_{ID}, h_{ID}, x_{ID}, s_{ID})$ and returns h_{ID}.

Partial-Private-Extract(ID): If $ID \neq ID^*$, the algorithm $\mathcal{F}$ invokes *Create(ID)* to obtain $(ID, P_{ID}, R_{ID}, h_{ID}), x_{ID}, s_{ID})$ and returns s_{ID}. If $ID = ID^*$, the algorithm $\mathcal{F}$ stops.

Key-Gen(ID): $\mathcal{F}$ responses the previously $pk_{ID} = (P_{ID}, R_{ID})$ if the ID has already stored in L_C. If not, $\mathcal{F}$ invokes *Create(ID)* to obtain $(ID, P_{ID}, R_{ID}, h_{ID}), x_{ID}, s_{ID},)$ and returns pk_{ID}. If $ID = ID^*$, the algorithm $\mathcal{F}$ stops. If $ID \neq ID^*$, $\mathcal{F}$ replies with x_{ID}.

Public-Key-Replacement($ID, pk_{ID'}$): L_R stores all tuples with the content of $(ID, r_{ID}, R_{ID}, x_{ID}, P_{ID})$. When $\mathcal{A}_2$ queries $(ID, pk_{ID'})$, $\mathcal{F}$ computes $P_{ID} = P_{ID'}$, $R_{ID} = R_{ID'}$, and calculates $s_{ID} = \perp$, $x_{ID} = x_{ID'}$, and then adds $(ID, rID', P_{ID'}, x_{ID'}, R_{ID'})$ to the hash list L_R.

H_2-query: $\mathcal{A}_2$, holding ID, queries H_2 to obtain h_{ID}, $\mathcal{F}$ selects $h \in \mathbb{Z}_q$ and computes $h = H_2(m||index||ID)$, $(m, ID, P_{ID}, R_{ID}, h)$ and outputs h.

Sign(ID, m): $\mathcal{A}_2$ asks the query of (ID, m). If $ID \neq ID^*$, the algorithm $\mathcal{F}$ executes as the steps before. Otherwise, the algorithm $\mathcal{F}$ computes $s = x_{ID} \cdot m \cdot T + h \cdot s_{ID} = x_{ID} \cdot m \cdot T + h \cdot r_{ID} + h_{ID} \cdot x \cdot h$ and output it.

Create oracle fails if the inconsistency occurs with probability at most q_h/n under $H_1(ID, R_{ID}, P_{ID})$. The successful of the oracle is q_c times with probability $(1 - q_h/n)^{q_c}$ at least $1 - \frac{q_h q_c}{n}$. H_2 oracle fails if the inconsistency occurs at most q_h/n under $H_2(m||index||ID)$. The successful of oracle is q_h times with probability $(1 - q_h/n)^{q_n}$ at least $1 - \frac{q_h^2}{n}$. Furthermore, the probability of $ID = ID^*$ is $1/q_c$. Combining them together, the successful probability is less than $\frac{1}{q_c}(1 - \frac{q_h q_c}{n})(1 - \frac{q_h^2}{n})$.

In all, if $\mathcal{A}_1$ or $\mathcal{A}_2$ can successfully forge a valid signature, $\mathcal{F}$ can solve DLP, which is **contradicted** to the assumption of DLA.

Theorem 5. Our scheme is anti-replace attack where no adversary $\mathcal{A}$ (database server) can successfully pass the verification with an invalid proof.

Proof 5. Passing the verification requires all the strings, including signatures, valid files, challege messages are corrected. Corrupted or non-updated files cannot pass the verification. Replace attack means the sever uses an invalid string to replaces the original correct ones, such as $\{s_l, h_l, m_l\}$ to replace s_j, h_j, m_j and proof as $S' = v_j \cdot s_l + \sum_{i \in I, i \neq j} v_i \cdot s_i, H' = v_j h_l + \sum_{i \in I, i \neq j} v_i \cdot h_i, \mu' = v_j \cdot m_l + \sum_{i \in I, i \neq j} v_i \cdot m_i$. Then, we have relation $e(S', P) = e(v_j \cdot (h_l - h_j) \cdot h_{ID} + \sum_{i \in I} v_i \cdot h_i \cdot h_{ID}, P_{pub})$ which is proven in Eq. (5) as follows.
When and only when $h_l - h_j = 0$, the equation is succeed, which means a server can pass the verification. However, $h_l - h_j$ cannot reach to 0 due to the collision resistance of hash functions. Equally, the database server cannot pass the verification process.

Theorem 6. Our scheme is zero-knowledge where no adversary $\mathcal{A}$ (public verifier) can successfully obtain useful information of outsourced files during the verification.

Proof 6. In the verification, a public verifier have the access of messages $chall = \{\ell, I\}$ and $Por = (S, H, \mu)$. But useful information cannot be recovered from Por due to its form of $S = \sum_{i \in I} v_i \cdot s_i$, $H = \sum_{i \in I} v_i \cdot h_i$, $\mu = \sum_{i \in I} v_i \cdot m_i$. The verifier cannot obtain s_i, h_i, m_i from Equation though he can compute $i \in I, v_i = \ell^i \mod q$. The probability of guessing a correct value depends on its unknown $|I|$ variables. They needs at least $q^{|I|}$ times to test for the equation of $S = \sum_{i \in I} v_i \cdot s_i$, $H = \sum_{i \in I} v_i \cdot h_i$, $\mu = \sum_{i \in I} v_i \cdot m_i$, where only one solution is the correct. Equally, the probability of obtaining s_i, h_i, m_i is $q^{-|I|}$ is pretty low. Furthermore, it is impossible for the verifier to extract m_i from h_i as the hash function is irreversible. Therefore, the verifier cannot obtain any useful information in the verification.

5.3 Discussion on Anti-attack

Our scheme can resist main attacks by the following way. Firstly, we back to why attackers may threaten the system. Based on a hybrid architecture, the attacks of the system comes from three aspects: underlying cryptosystems, blockchain systems, and their combinations. The cryptographic building blocks may be compromised by attacks due to its inherent design drawbacks; the blockchain system may confront several mainstream attacks; and their combinations may cause some unpredictable logic errors like deliberately hiding a negative value in ciphertext when sending transactions.

Then, to evaluate these vulnerabilities, we observe that the prior two factors closely rely on their assumptions. For the Pailler cryptosystem, the securities and properties are formally proved by its original paper [15]. Here, we just present the correctness of its basic scheme, as shown in Theorem 1. For the blockchains system, it is acknowledged that the systems confront multiple types of attacks including double-spending attacks, selfish attacks *etc.* We simplify the system model and capture the main reason that the permissionless blockchains may attract more attackers. Thus, we adopt the permissioned blockchain to ensure only a small group of members are allowed to maintain and manage the chains.

Next, for the last factor related to their combinations, we identified three types of attacks as general cases, namely *inconsistency attack*, *inconsistency attack*, *replace attack*, and *exposure attack*. The inconsistency attack means the outputs of encrypted bills are inconsistent with the initial inputs. An attacker may forge the bills by sending negative values in transactions. We utilize two additional non-interactive zero-knowledge proof techniques to guarantee the consistency: Theorem 2 ensures the input (sum) values are equal to the output (sum) values; Theorem 3 ensures the encrypted values are positive. These two theorems prevent the scheme from being attacked by the inconsistency attack. The replace

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.2020.3016721, IEEE Internet of Things Journal

$$
\begin{aligned}
e(S', P) &= e(v_j \cdot s_l + \sum_{i \in I, i \neq j} v_i \cdot s_i, P) \\
&= e(v_j \cdot (x_{ID} \cdot m_l \cdot T + h_j \cdot s_{ID}) + \sum_{i \in I, i \neq j} v_i \cdot (x_{ID} \cdot m_i \cdot T + h_i \cdot s_{ID}), P) \\
&= e(v_j \cdot (x_{ID} \cdot m_l \cdot T + h_j \cdot (r_{ID} + h_{ID} \cdot x)) + \sum_{i \in I, i \neq j} v_i \cdot (x_{ID} \cdot m_i \cdot T + h_i \cdot (r_{ID} + h_{ID} \cdot x)), P) \\
&= e(v_j \cdot x_{ID} \cdot m_l \cdot T + \sum_{i \in I, i \neq j} v_i \cdot x_{ID} \cdot m_i \cdot T, P) \cdot e(v_j \cdot h_l \cdot r_{ID} + \sum_{i \in I, i \neq j} v_i \cdot h_i \cdot r_{ID}, P) \\
&\cdot e(v_j \cdot h_l \cdot h_{ID} \cdot x + \sum_{i \in I, i \neq j} v_i \cdot h_i \cdot h_{ID} \cdot x, P) \\
&= e(v_j \cdot m_l \cdot T + \sum_{i \in I, i \neq j} v_i \cdot m_i \cdot T, P_{ID}) \cdot e(v_j \cdot h_l + \sum_{i \in I, i \neq j} v_i \cdot h_i, R_{ID}) \cdot e(v_j \cdot h_l \cdot h_{ID} + \sum_{i \in I, i \neq j} v_i \cdot h_i \cdot h_{ID}, P_{pub}) \\
&= e(T, \mu' \cdot P_{ID}) \cdot e(v_j \cdot (h_l - h_j) + \sum_{i \in I} v_i \cdot h_i, R_{ID}) \cdot e(v_j \cdot (h_l - h_j) \cdot h_{ID} + \sum_{i \in I} v_i \cdot h_i \cdot h_{ID}, P_{pub})
\end{aligned} \tag{5}
$$

Table 1
Computation cost on personal end

Complexity	Unit of Time
Key generation†	$(i+2)\tau_m$
Encryption†	$2i\tau_M + 4i\tau_E$
Verification†	$(2i+3)\tau_m + (7i+8)\tau_M + (12i+14)\tau_E + 2\tau_H$
Decryption†	$2\tau_m + 2i\tau_E$
ProofGen‡	$c\tau_E + 3c\tau_M$
ProofVerify‡	$4\tau_P + c\tau_E + (3c+4)\tau_M + \tau_H$
Common auditing*	$m[4\tau_P + c\tau_E + (3c+4)\tau_M + \tau_H]$
Batch auditing*	$4\tau_P + mc\tau_E + (3mc+4)\tau_M + m\tau_H$

attack means an adversary can forge an invalid proof to pass the verification. Theorem 5 provides proof on how to defend this type of attack. The *exposure attack* represents that the sensitive information of users may be unconsciously exposed to attackers or unrelated organizations like governments. Theorem 2, Theorem 3, and Theorem 6 employ the technique of zero-knowledge proof to mitigate the issue.

Therefore, We prove the theoretical-securities in Theorems 1-6, with additional system-level security assumptions based on the blockchain system. As a summary, our scheme can prevent mainstream attacks.

5.4 Performance analysis

We provide the efficiency analysis of each sub-algorithms to show the performance of our scheme. We evaluate the performance by analysing the calculation overheads. The notations are used to describe the complexity cost of multiplication operation τ_m, modular multiplication τ_M, modular exponentiation τ_E, and hash function operation τ_H, pair operation τ_P, respectively. We use the †, ‡ to represent corresponding sub-algorithms which are separately executed on personal end and bulk end. Additionally, to highlight the difference between the common auditing and the batch auditing, we provide the computation overhead with the notation $\star$. The complexities of each sub-algorithms are shown in Table 1.

For the personal end, the complexities cover four main sub-algorithms. The verfication process spends the most time in protocol, much more than processes of keygen, encryption, decryption, which take times at the same level. For the bulk end, two essential sub-algorithem are considered. For *ProofGen*, a public verifier sends the messages $chall = \ell, I$ to the database sever. Then database generates the proof $Pro = (S, H, \mu)$ and sends it back to the verifier. The computation overhead to generate a proof is $c\tau_E + 3c\tau_M$. The cost of verifying a proof is $4\tau_P + c\tau_E + (3c+4)\tau_M + \tau_H$. The communication cost of the challenge messages is $|q| + c|n|bits$, and the complexity of response is $3|q|bits$, and n is the shares of blocks, $|q|$ represents the length of an element in $\mathbb{Z}_q$, $|p|$ is the length of an element in $\mathbb{Z}_p$ and $|id|$ represents the length of identifies. When multiple files are audited at the same time, batch auditing is reduce the costs and becomes much more efficient than a common auditing process. For comparison, the batch auditing is faster than a common auditing on n level. Here, m represents the number of auditing files.

6 Related Work

Public Auditing. Public auditing is essential to the correctness and integrity of stored files in centralized servers. To guarantee the neutrality and honesty, a third trusted authority is necessary for the schemes, which is called third-party auditor (TPA). However, sending transparent files to the sever center is not reliable due to the potential leakage of sensitive data. Users may require their files are protected with high privacy. The confidential information is, as a consequence, sent to the servers. Therefore, the technique of privacy protection is considered in public auditing protocol. Wang *et al.* [6] proposes a scheme by employing the ring signature combined with a homomorphic authenticator to prevent the identities of users from exposure. Yu *et al.* [16] protects the public data from being exposed to the verifiers by utilizing the technique of zero-knowledge proof. Besides these, the efficiency is also taken into account by the auditing schemes. Hwang *et al.* [11] proposes a batch auditing scheme to improve the efficiency of processing large scale data. Other schemes such as [9], [17], [18] provides their customized solutions which target to the similar requirements.

Privacy in Blockchain. Traditionally, Bitcoin, as the first blockchain prototype, provides a pseudonym for the trading parties through cryptographic algorithms. however, the constructed anonymity as the only privacy guarantee is weak security since the attackers can analyse the whole trade graph whenever they link the addresses and entities. Equally, when the user's actual identity and public key are binded together, the transaction history of the user will be easily read. In order to improve privacy, Heilman proposes E-cash [19] by employing the blind signature and crypto protocol to improve the anonymity and fairness of bitcoin. In 2013, Eli proposed the concept of ZeroCoin [20] to enhance the privacy of bitcoin. The scheme designed a distributed "cleaning" mechanism through zero knowledge proof (ZKP) to confuse inputs and outputs. In 2014, on the basis of the previous research, Miers *et al.* further proposed a more complete system called ZeroCash [21], which provides an enhanced privacy. In the same year, Bonneau *et al.* put forward MixCoin [22], preventing from potential attacks through currency accountability mechanisms. The scheme improved the privacy by providing a classical mixed communication similar scheme. Adam Back *et al.* [23] proposed the concept of Confidential Transaction (CT) on the Bitcointalk in 2013. As the core developer of the bitcoin community, Gregory Maxwell [24] implemented the homomorphic operation through Perdersen commitment mechanism. Menero [25], as the currency supported by CryptoNote, confused the input participants through ring signature cryptography scheme. The updated 2.0 version [26] has improved the calculation rate of ring signature through accumulator. And the usability is significantly better than before.

7 Conclusion

In this paper, we propose a pAuditChain which realize the practical demands in industrial electricity supplies. The system achieves the auditing function both for individual users with little data and authorities/organisations such as government with bulk data. pAuditChain employs homomorphic encryption cooperated with blockchain structure to seal the secret data like bills, smoothly balancing the data auditing and privacy preserving requirements. The system also employes certificateless public verification method to achieve the auditing function while maintaining the high performance. The system is equipped with a series of theoretical analysis, especially on the privacy preserving and auditing processes. To the best of our knowledge, pAuditChain is the first solutions to provide the privacy protection of bills generated by IoT smart meters with a final auditing. Also, our approach could be further applied to other forms of IoT devices based on bills.

References

[1] Alex. (2016) Electricity tecnical report @ONLINE. [Online]. Available: https://www.theguardian.com/technology/2016/dec/29/smart-electricity-meters-dangerously-insecure-hackers

[2] (2018) Silver spring networks @ONLINE. [Online]. Available: https://www.silverspringnet.com

[3] Alex. (2018) Gridnet @ONLINE. [Online]. Available: https://www.grid-net.com

[4] (2018) Smart meters technical report @ONLINE. [Online]. Available: http://www.smartmeters.vic.gov.au/about-smart-meters/reports-and-consultations/advanced-metering-infrastructure-cost-benefit-analysis/3.-technology-deployed-in-victoria

[5] (2018) Smart meters security report @ONLINE. [Online]. Available: https://www.smh.com.au/business/consumer-affairs/smart-meters-questioned-for-their-security-and-privacy-safeguards-20170426-gvsoxx.html

[6] B. Wang, B. Li, and H. Li, "Oruta: Privacy-preserving public auditing for shared data in the cloud," *IEEE transactions on cloud computing*, vol. 2, no. 1, pp. 43–56, 2014.

[7] L. Huang, G. Zhang, and A. Fu, "Privacy-preserving public auditing for dynamic group based on hierarchical tree," *Journal of Computer Research and Development*, vol. 53, no. 10, 2016.

[8] ——, "Privacy-preserving public auditing for non-manager group," in *2017 IEEE International Conference on Communications (ICC)*. IEEE, 2017, pp. 1–6.

[9] C. Wang, S. S. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for secure cloud storage," *IEEE Transactions on computers*, vol. 62, no. 2, pp. 362–375, 2013.

[10] L. Huang, G. Zhang, and A. Fu, "Certificateless public verification scheme with privacy-preserving and message recovery for dynamic group," in *Proceedings of the Australasian Computer Science Week Multiconference*. ACM, 2017, p. 76.

[11] M.-S. Hwang, T.-H. Sun, and C.-C. Lee, "Achieving dynamic data guarantee and data confidentiality of public auditing in cloud storage service," *Journal of Circuits, Systems and Computers*, vol. 26, no. 05, p. 1750072, 2017.

[12] S. S. Al-Riyami and K. G. Paterson, "Certificateless public key cryptography," *Asiacrypt*, vol. 2894, no. 2, pp. 452–473, 2003.

[13] Q. Wang, B. Qin, J. Hu, and F. Xiao, "Preserving transaction privacy in bitcoin," *Future Generation Computer Systems*, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X17318393

[14] F. Boudot, "Efficient proofs that a committed number lies in an interval," in *Advances in Cryptology — EUROCRYPT 2000*, B. Preneel, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000.

[15] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 1999.

[16] Y. Yu, M. H. Au, Y. Mu, S. Tang, J. Ren, W. Susilo, and L. Dong, "Enhanced privacy of a remote data integrity-checking protocol for secure cloud storage," *International Journal of Information Security*, vol. 14, no. 4, pp. 307–318, 2015.

[17] Y. Yu, L. Xue, M. H. Au, W. Susilo, J. Ni, Y. Zhang, A. V. Vasilakos, and J. Shen, "Cloud data integrity checking with an identity-based auditing mechanism from rsa," *Future Generation Computer Systems*, vol. 62, pp. 85–91, 2016.

[18] M. S. Kiraz, I. Sertkaya, and O. Uzunkol, "An efficient id-based message recoverable privacy-preserving auditing scheme," in *Privacy, Security and Trust (PST), 2015 13th Annual Conference on*. IEEE, 2015, pp. 117–124.

[19] E. Heilman, F. Baldimtsi, and S. Goldberg, "Blindly signed contracts: Anonymous on-blockchain and off-blockchain bitcoin transactions," in *International Conference on Financial Cryptography and Data Security*. Springer, 2016, pp. 43–60.

[20] I. Miers, C. Garman, M. Green, and A. D. Rubin, "Zerocoin: Anonymous distributed e-cash from bitcoin," in *Security and Privacy (SP), 2013 IEEE Symposium on*. IEEE, 2013, pp. 397–411.

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

[22] J. Bonneau, A. Narayanan, A. Miller, J. Clark, J. A. Kroll, and E. W. Felten, "Mixcoin: Anonymity for bitcoin with accountable mixes," in *International Conference on Financial Cryptography and Data Security*. Springer, 2014, pp. 486–504.

[23] A. Back, "Bitcoins with homomorphic value (validatable but encrypted)," *Bitcointalk (accessed 1 May 2015) https://bitcointalk.org/index. php*, 2013.

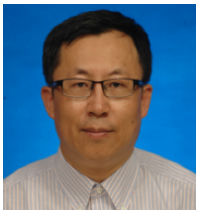
[24] G. Maxwell, "Confidential transactions," in *Bitcoin forum*, 2015.

[25] S. Noether, "Ring signature confidential transactions for monero." *IACR Cryptology ePrint Archive*, vol. 2015, p. 1098, 2015.

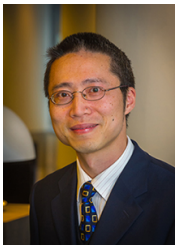
[26] S.-F. Sun, M. H. Au, J. K. Liu, and T. H. Yuen, "Ringct 2.0: a compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency monero," in *ESORICS*. Springer, 2017, pp. 456–474.

Qin WANG is currently pursuing the Ph.D. in Swinburne University of Technology, Australia, cooperated with CSIRO Data61, Australia. He received his M.S degree in Beihang University, Beijing, China in 2018, and B.S. degree in Northwestern Polytechnical University, Xi'an, China in 2015. His research interests include blockchain technology, and cryptography.

Longxia HUANG received the Ph.D. degree in Computer Science and Technology from Nanjing University of Science and Technology, Jiangsu, China, in 2019. She was a Visiting Scholar with the Deakin University, Melbourne, Australia, in 2018 and 2019.She is currently a Lecture with the School of Computer Science & Communication Engineering, Jiangsu University, Jiangsu, China. Her current research interests include blockchain technology, privacy protection, data sharing and cloud computing.



Shiping CHEN is a principal research scientist at CSIRO Data61. He also holds an adjunct A/Professor title with the University of Sydney and the University of New South Wales (UNSW) through supervising PhD students. He supervised 15 PhD students in the past 12 years, including 7 PhD successfully completed. He has been working on distributed systems for over 20 years with focus on performance and security. He published 150+ research papers in these areas. His work is well reorganized via paper citations, software downloading, technology licensing and academic awards, such as the ACM SIGSOFT Distinguished Paper Award on ICSE-2018. He is also actively involved in software engineering and service/cloud computing research communities through publications, journal editorships and conference Chairs/TPC services, including WWW, ICSOC, IEEE ICBC, IEEE ICWS/SCC/CLOUD etc. His current research interests include application security, blockchain and service collaboration. He is a senior member of the IEEE.



Yang XIANG received his PhD in Computer Science from Deakin University, Australia. He is currently a full professor and the Dean of Digital Research & Innovation Capability Platform, Swinburne University of Technology, Australia. His research interests include cyber security, which covers network and system security, data analytics, distributed systems, and networking. In particular, he is currently leading his team developing active defense systems against large-scale distributed network attacks. He is the Chief Investigator of several projects in network and system security, funded by the Australian Research Council (ARC). He has published more than 200 research papers in many international journals and conferences. He served as the Associate Editor of IEEE Transactions on Dependable and Secure Computing, IEEE Internet of Things Journal, IEEE Transactions on Computers, IEEE Transactions on Parallel and Distributed Systems, and the Editor of Journal of Network and Computer Applications. He is the Coordinator, Asia for IEEE Computer Society Technical Committee on Distributed Processing (TCDP). He is a Senior Member of the IEEE.