Copyright ⓒ 2021 KSII

# A Privacy-preserving Data Aggregation Scheme with Efficient Batch Verification in Smart Grid

**Yueyu Zhang[1,2], Jie Chen[2,3*], Hua Zhou[2], and Lanjun Dang[1]**
[1] School of Cyber Engineering, Xidian University
Xi'an, Shaanxi, 710071, China
[e-mail: yyzhang@xidian.edu.cn, ljdang@mail.xidian.edu.cn]
[2] State Key Laboratory of Integrated Service Networks, Xidian University
Xi'an,Shaanxi, 710071, China
[e-mail: jchen@mail.xidian.edu.cn, iris_zhouily@163.com]
[3] Cryptography Research Center, Xidian University
Xi'an, Shaanxi, 710071, China
[e-mail: jchen@mail.xidian.edu.cn]
*Corresponding author: Jie Chen

*Received May 16, 2016; revised May 8, 2020; accepted December 30, 2020; published February 28, 2021*

## Abstract

This paper presents a privacy-preserving data aggregation scheme deals with the multidimensional data. It is essential that the multidimensional data is rarely mentioned in all researches on smart grid. We use the Paillier Cryptosystem and blinding factor technique to encrypt the multidimensional data as a whole and take advantage of the homomorphic property of the Paillier Cryptosystem to achieve data aggregation. Signature and efficient batch verification have also been applied into our scheme for data integrity and quick verification. And the efficient batch verification only requires 2 pairing operations. Our scheme also supports fault tolerance which means that even some smart meters don't work, our scheme can still work well. In addition, we give two extensions of our scheme. One is that our scheme can be used to compute a fixed user's time-of-use electricity bill. The other is that our scheme is able to effectively and quickly deal with the dynamic user situation. In security analysis, we prove the detailed unforgeability and security of batch verification, and briefly introduce other security features. Performance analysis shows that our scheme has lower computational complexity and communication overhead than existing schemes.

***Keywords:*** Batch Verification, Homomorphic Encryption, Multidimensional Aggregation, Privacy Preserving, Smart Grid

This research was supported by the Natural Science Foundation of China (U1736111) and National Cryptography Development Fund (MMJJ20180219).

**http://doi.org/10.3837/tiis.2021.02.013** **ISSN : 1976-7277**

## 1. Introduction

With the development of smart grid technology, smart grid has increasingly shown its importance. Compared with the centralized one-way transmission of traditional grid, smart grid makes a feature of decentralized two-way transmission as shown in **Fig. 1** and is aimed at providing improved reliability, efficiency and sustainability, consumer involvement and security [1]. In smart grid, every user is equipped with a smart meter. On one hand, the smart grid needs to collect real-time information from smart meters for adjusting power supplies or changing electricity price etc. On the other hand, the aggregator is not supposed to know the personal detailed electricity consumption information on each user because the detailed electricity consumption information may leak the user's behavior information [2]. Therefore, there are two important tasks for security in the smart grid [3]. One is to hide the data privacy of the individual smart meter readings. At the same time, the other one is to allow the aggregator to obtain the overall electricity consumption information about all users.

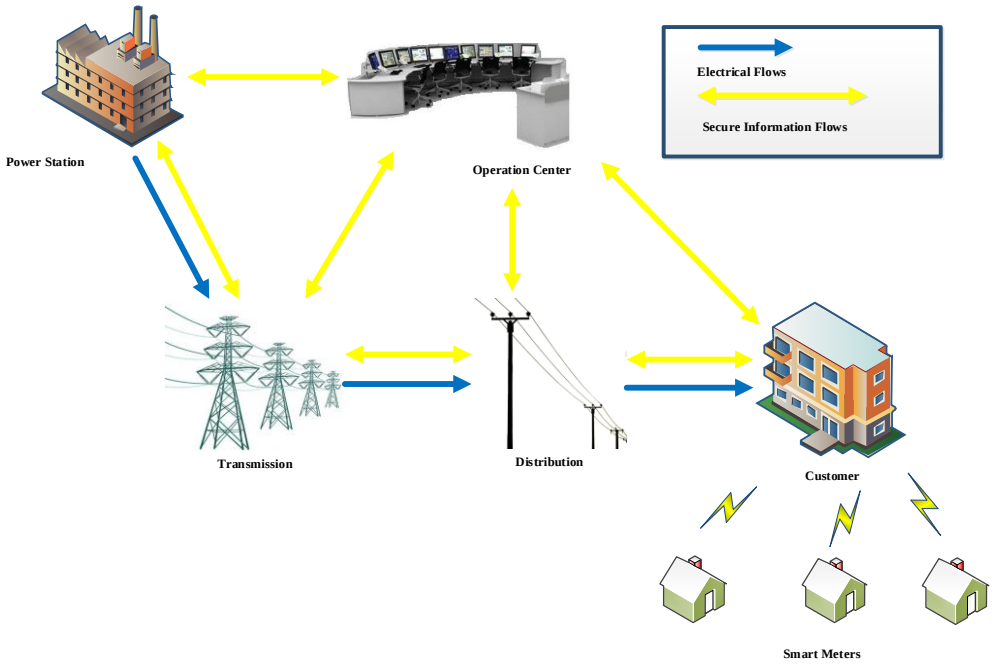



**Fig. 1.** The architecture of smart grid

To solve the two important tasks in smart grid, homomorphic Paillier encryption technique [4] can be applied. So the aggregator can perform the aggregation operation without decrypting the encrypted data of users. Such homomorphic encryption has been applied in many existing data aggregation schemes [1, 5-12]. Most of the aggregation schemes focused on one dimensional data [5-12], while Lu et al.'s scheme [1] focuses on multidimensional user data which is rarely mentioned in existing approaches. But the scheme of [1] has one drawback that it can't resist internal attackers. That is, if the operation authority (OA) is compromised by attackers, users' data will be revealed. And although they adopte the batch verification technique to reduce authentication cost, the time consuming of their pairing operations is still proportional to the number of users. Fan et al. proposed a privacy-enhanced data aggregation

scheme [13]. In [13], each user embeds a blinding factor into their ciphertext to protect their data from internal attackers; and the aggregator also holds one that makes the sum of all these blinding factors is 0. This kind of blinding factor technique has been applied in several works [6, 14-17]. But they all share one major drawback that they are not tolerant to smart meter failures. If some users fail to report their data, the aggregator will obtain the wrong aggregation result as the sum of the blinding factors is no longer 0. A privacy-preserving data aggregation scheme is proposed by Chen et al. [9]. In [9], TA takes charge of distributing the blinding factors to the aggregator and users. If some users' smart meters don't work, the aggregation scheme can still work well by the help of TA. But the communication overhead is significant in [9] and it cannot get message authentication. Based on above descriptions, this paper presents a privacy-preserving data aggregation scheme that can deal with the multidimensional data. System model includes a trusted authority (TA), an untrusted aggregator and users. We use the Paillier Cryptosystem and blinding factor technique to encrypt the multidimensional data as a whole and take advantage of the homomorphic property of the Paillier Cryptosystem to achieve data aggregation. Signature and efficient batch verification have also been applied into our scheme for data integrity and quick verification. And the efficient batch verification only requires 2 pairing operations. Our scheme also supports fault tolerance which means that if some users' smart meters don't work, our scheme can still work well with the help of TA. Furthermore, we consider the situation that the batch verification may fail. If the batch verification fails, we can use the technique proposed in [18] to quickly find the invalid signatures. In addition, we give two extensions of our scheme, one is that our scheme can be used to compute a fixed user's time-of-use electricity bill. The other is that our scheme is able to effectively and quickly deal with dynamic user situation. In security analysis, we prove the unforgeability and batch verification security in details and explain that our scheme can also resist external attackers and internal attackers. Through performance analysis, our scheme has lower computational complexity and communication overhead than existing schemes.

The remainder of this paper is organized as follows. The system model and preliminaries are introduced in Section 2. Our detailed scheme and the extensions of our scheme are given in Section 3 and Section 4, respectively. In Section 5, the security and performance analysis of our scheme are discussed. Lastly, we draw conclusions in Section 6.

## 2. Simplified System Model and Preliminaries

### 2.1 Simplified System Model

We consider simplified smart grid architecture in our system model, which includes a number of users, an aggregator and a trusted authority (TA) as shown in **Fig. 2**. This paper mainly deals with how to report users' data to the aggregator in a privacy-preserving way with the signature aggregation and batch verification.

- Trusted Authority (TA): TA is a trusted entity in our system model which belongs to some independent organizations like Regional Transmission Organizations (RTO) or Independent System Operators (ISO). In the initialization phase, TA selects blinding factors and sends them to the aggregator and each user. Besides that, if some smart meters cannot report the real-time data to the aggregator, TA can help to make the aggregation process to continue normal execution. We assume that TA cannot be compromised by any strong adversaries.
- Aggregator: The aggregator is a powerful entity who is curious about users' electricity consumption data. In our system model, the aggregator might be compromised by the

outside adversaries, thus it is untrusted. During the initialization phase, the aggregator generates secret values and publishes public information. During the aggregation phase, the aggregator can use the batch verification to verify users' signatures and get users' total electricity consumption information without knowing each user's.

- Users $\mathrm{U}=\{U_1,U_2,\ldots,U_n\}$: $\mathrm{U}$ represents user set. If there are *n* users, $\mathrm{U}=\{U_1,U_2,\ldots,U_n\}$. Each user $U_i \in \mathrm{U}$ is equipped with a smart meter that can record the real-time electricity consumption information. These real-time data will be reported to the aggregator in a certain period, i.e. every 15 minutes. Sometimes smart meters may malfunction i.e. they may stop reporting for a while or reset it later.

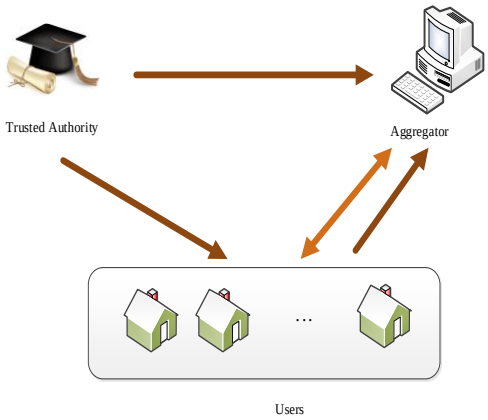



**Fig. 2.** System model

## 2.2 Bilinear Pairing Setting

We define two cyclic multiplicative groups $\{\mathrm{G}_1,\mathrm{G}_2\}$ with the same prime order *q*. $g$ is a generator of $\mathrm{G}_1$. $\mathrm{G}_1$ and $\mathrm{G}_2$ possess a nondegenerated and efficiently computable bilinear map $e:\mathrm{G}_1\times\mathrm{G}_1\to\mathrm{G}_2$. The bilinear pairing contains the following features [13].

- Bilinearity: $e(P,P)\neq 1_{\mathrm{G}_2}$ and $e(P^a,Q^b)=e(P,Q)^{ab}\in\mathrm{G}_2$ for all $P,Q\in\mathrm{G}_1$, $a,b\in Z_q^*$.
- Nondegeneracy: There exists $P,Q\in\mathrm{G}_1$ such that $e(P,Q)\neq 1_{\mathrm{G}_2}$.
- Computability: There exists an efficient algorithm to compute $e(P,Q)$ for all $P,Q\in\mathrm{G}_1$.

## 2.3 Paillier Cryptosystem

The Paillier Cryptosystem [4] can acquire the homomorphic properties. The Paillier cryptosystem specifically contains key generation, encryption and decryption [1].

- Key Generation: According to the security parameter $k$, two large prime numbers $p,q$ are first selected, where $|p|=|q|=k$. Then the RSA modulus are computed with $N=pq$ and $\lambda=lcm(p-1,q-1)$. Define a function $L(u)=\dfrac{u-1}{N}$, after choosing a generator $g\in Z_{N^2}^*$, $\mu=\left(L\left(g^{\lambda} \bmod N^2\right)\right)^{-1} \bmod N$ is further calculated. Then, the public key is $pk=(N,g)$, and the corresponding private key is $sk=(\lambda,\mu)$.

- Encryption: Given a message $m \in Z_N$, select a random number $r \in Z_N^*$. The ciphertext of $m$ is $c = E(m) = g^m \cdot r^N \bmod N^2$.
- Decryption: Given the ciphertext $c \in Z_{N^2}^*$, the corresponding message can be recovered as $m = D(c) = L\left(c^\lambda \bmod N^2\right) \cdot \mu \bmod N$.

In addition, the Paillier encryption can be proved to satisfy the chosen plaintext security, and the correctness and security can be referred to [4].

### 2.4 Small Exponent Test

If someone needs to verify the equations set $Y_i = g^{X_i}$ ( $i = 1,2,\ldots,n$ ), where $g$ is a generator for a group of prime order $q$. One can calculate and check if the equation $\prod_{i=1}^{n} Y_i = g^{\sum_{i=1}^{n} X_i}$ is true. If there are such two pairs ($x_1$, $y_1$) and ($x_2$, $y_2$), their product can be verified correctly, but each verification is not correct, for example, by submitting the pairs $(X_1 - \alpha, Y_1)$ and $(X_2 + \alpha, Y_2)$ for any $\alpha$ [19]. Because of the above reasons, the small exponent test method is proposed in [20], which will be applied to pairings in the following section.

Small exponent test [20]: choose a small exponent $\delta_i$ of $\ell_b$ bits and compute $\prod_{i=1}^{n} Y_i^{\delta_i} = g^{\sum_{i=1}^{n} X_i \delta_i}$. The probability is $2^{-\ell_b}$ when a bad pair is accepted. Choose the size of $\ell_b$ according to the trade-off between efficiency and security.

### 2.5 Notation

Some notations are defined as follows.

$t$: the time when the aggregator needs to aggregate the power usage data.

$\mathrm{U}$: the user set.

$U_i$: the users in the neighborhood, where $i = 1,2,\ldots,n$.

$ID_i$: the identifier of $U_i$.

$\pi_i$: the $U_i$'s blinding factor.

$\pi_0$: the blinding factor of aggregator.

$N, g_{Ag}$: the public key of aggregator.

$\lambda, \mu$: the private key of aggregator.

$X_i$: the $U_i$'s private key.

$Y_i$: the $U_i$'s public key.

## 3. Proposed Scheme

### 3.1 Initialization Phase

**Aggregator**: First the aggregator calculates the Paillier Cryptosystem's public key $\left(N = p_1 q_1, g_{Ag}\right)$, and the corresponding private key $(\lambda, \mu)$, where $p_1$, $q_1$ are two large primes,

$g_{Ag}$ is a generator of $Z^*_{N^2}$, $\lambda = lcm(p_1 - 1, q_1 - 1)$ and $\mu = \left(L\left(g_{Ag}{}^{\lambda} \bmod N^2\right)\right)^{-1} \bmod N$. Assume that there needs to report $l$ types of electricity usage data in total. Then the aggregator needs to choose a sequence $\vec{a} = (a_1, a_2, \ldots, a_l)$ where $a_i \in Z^+$, for $i = 1,2,\ldots,l$. And the aggregator calculates $(g_1, g_2, \ldots, g_l)$ where $g_i = g_{Ag}^{a_i}$, for $i = 1,2,\cdots,l$. Lastly, the public information $\{N, (g_1 \cdots g_l)\}$ is published by the aggregator, and $\{\lambda, \mu\}$ is kept as secrets.

**TA**: Choose the security parameter $k$, TA runs $Gen(k)$ to generate $(q, g, G_1, G_2, e)$. TA also needs to select $(n+1)$ blinding factors $\{\pi_0, \pi_1, \cdots, \pi_n\}$ at random such that $\pi_0 + \pi_1 + \cdots + \pi_n \equiv 0 \bmod N$. Firstly, $n$ random numbers $\{\pi_1, \cdots, \pi_n\}$ ($\pi_i \in Z_N$, $i = 1,\ldots,n$) are generated by running pseudorandom generators, and $\pi_0 = -(\pi_1 + \cdots + \pi_n) \bmod N$ is computed. Actually, the size should not be less than 1024 bits for each blinding factor. Lastly, TA respectively sends $\pi_0$ to the aggregator and $\pi_i$ to $U_i$ for each $i = 1,2,\ldots,n$. In addition, TA need to select three secure hash functions $H_1$, $H_2$, $H_3$ where $H_1 : \{0,1\}^* \to G_1$, $H_2 : \{0,1\}^* \to Z^*_N$, and $H_3 : \{0,1\}^* \to Z^*_q$.

$U_i$: $U_i$ selects a random number $X_i \in Z^*_q$ and computes the corresponding value $Y_i$ where $Y_i = g^{X_i}$. Then $X_i$ is $U_i$'s private key and $Y_i$ is its public key.

### 3.2 Report Phase

Each user $U_i \in U$ collects $l$ types of data ($d_{i1}$, $d_{i2}$,…, $d_{il}$) by the smart meter in a period time so as to achieve the real-time users' electricity consumption data. Then $U_i$ performs the following steps.

(1) After getting the $l$ types of data ($d_{i1}$, $d_{i2}$,…, $d_{il}$) from the smart meter, $U_i$ computes the ciphertext $CT_i = g_1^{d_{i1}} \cdot g_2^{d_{i2}} \cdot \cdots \cdot g_l^{d_{il}} \cdot \left(H_2(t)\right)^{\pi_i} \bmod N^2$ according to Paillier Cryptosystem.

(2) $U_i$ first calculates two values $h_i$, $W$, where $h_i = H_3(CT_i)$, $W = H_1(t)$ and then computes $V_i = W^{X_i h_i}$. Finally the signature $\sigma_i = V_i$.

(3) $U_i$ reports the ciphertext and signature $\{\sigma_i, CT_i\}$ to the aggregator.

### 3.3 Aggregation Phase

After receiving total $n$ reports $\{\sigma_i, CT_i\}$ from the users where $i = 1,2,\ldots,n$, the aggregator calculates $h_i = H_3(CT_i)$ for $i = 1,2,\ldots,n$ and $W = H_1(t)$ and performs the following steps.

(1) Firstly, the aggregator verifies all signatures by checking whether $e(g, \sigma_i) = e(W, Y_i^{h_i})$. In order to improve the efficiency of verification, the aggregator can perform the batch verification by checking whether $e\left(g, \prod_{i=1}^{n} \sigma_i^{\delta_i}\right) = e\left(W, \prod_{i=1}^{n} \left(Y_i^{h_i}\right)^{\delta_i}\right)$ where $\delta_i$ is a random element in $Z^*_q$. Then the time-consuming pairing operation $e(\cdot,\cdot)$ can be reduced from $2n$ to 2 times.

(2) If the signatures are all valid in step 1, the aggregator computes $V = H_2(t)^{\pi_0} \prod_{i=1}^{n} CT_i \bmod N^2$ , where $V$ satisfies $V = g_{Ag}^{a_1 \sum_{i=1}^{n} d_{i1} + a_2 \sum_{i=1}^{n} d_{i2} + \cdots + a_l \sum_{i=1}^{n} d_{il}} \cdot H_2(t)^{\beta N} \bmod N^2$. This equation will be explained in Section 3 and 4. Here we take $M = a_1 \sum_{i=1}^{n} d_{i1} + a_2 \sum_{i=1}^{n} d_{i2} + \cdots + a_l \sum_{i=1}^{n} d_{il}$ and $R = (H_2(t))^{\beta}$. And we can see the report $V = g_{Ag}^{M} R^N \bmod N^2$ is a ciphertext of the Paillier Cryptosystem. Thus the aggregator can get $M$ by taking the decryption algorithm of the Paillier Cryptosystem. By taking the Algorithm 1 [1], the aggregator can recover the summation of the data with the different type ($D_1$, $D_2$,…, $D_l$), where $D_j = \sum_{i=1}^{n} d_{ij}$ ($j$=1, 2, …, $l$) represents the data aggregation value for type $j$.

The batch verification will fail when any of the signatures is invalid. In this case, we can use the technique proposed by Law and Matt [18] in 2007 for tracing the users who provide invalid signatures in a batch.

**Algorithm 1** [1]. Recover the aggregated report

1: **procedure**

**Input:** $\vec{a} = (a_1, a_2, \ldots, a_l)$ and $M$

**Output:** $(D_1, D_2, \ldots, D_l)$

2: Set $X_l = M$

3: for $n = l$ to 2 do

4: $X_{n-1} = X_n \bmod a_n$

5: $D_n = \dfrac{X_n - X_{n-1}}{a_n} = \sum_{i=1}^{n} d_{in}$

6: **end for**

7: $D_1 = X_1 = \sum_{i=1}^{n} d_{i1}$

8: **return** $(D_1, D_2, \ldots, D_l)$

9: **end procedure**

### 3.4 Correctness

**Signature Verification**: According to the bilinearity feature of the bilinear pairing mentioned in Section 2.2, we give the correctness verification as (1).

$$
\begin{aligned}
e\left(W,\prod_{i=1}^{n}\left(Y_i^{h_i}\right)^{\delta_i}\right) &= e\left(W,\prod_{i=1}^{n}\left(g^{x_i h_i}\right)^{\delta_i}\right) \\
&= \prod_{i=1}^{n} e\left(W, g^{x_i h_i \delta_i}\right) \\
&= \prod_{i=1}^{n} e\left(W^{x_i h_i \delta_i}, g\right) \\
&= \prod_{i=1}^{n} e\left(V_i^{\delta_i}, g\right) \\
&= e\left(\prod_{i=1}^{n} \sigma_i^{\delta_i}, g\right)
\end{aligned} \tag{1}
$$

**Ciphertext Decryption**: In the aggregation phase, if they are valid for all the signatures, the aggregator needs to compute the value $V$ for further decryption. We will now verify the correctness of the ciphertext decryption in our scheme. We use the feature of the blinding factors that $\pi_0 + \pi_1 + \cdots + \pi_n \equiv 0 \bmod N$ so the correctness verification is given as following (2).

$$
\begin{aligned}
V &= H_2(t)^{\pi_0} \prod_{i=1}^{n} CT_i \bmod N^2 \\
&= H_2(t)^{\pi_0} \prod_{i=1}^{n} g_1^{d_{i1}} \cdot g_2^{d_{i2}} \cdots\cdots g_l^{d_{il}} H_2(t)^{\pi_i} \bmod N^2 \\
&= \prod_{i=1}^{n} g_1^{d_{i1}} \cdot g_2^{d_{i2}} \cdots\cdots g_l^{d_{il}} H_2(t)^{\pi_0+\pi_1+\cdots+\pi_n} \bmod N^2 \\
&= g_1^{\sum_{i=1}^{n} d_{i1}} \cdot g_2^{\sum_{i=1}^{n} d_{i2}} \cdots\cdots g_l^{\sum_{i=1}^{n} d_{il}} \cdot H_2(t)^{\sum_{j=0}^{n} \pi_j} \bmod N^2 \\
&\xrightarrow{\sum_{j=0}^{n} \pi_j = 0 \bmod N \Rightarrow \sum_{j=0}^{n} \pi_j = \beta N \text{ for some } \beta} \\
&= g_1^{\sum_{i=1}^{n} d_{i1}} \cdot g_2^{\sum_{i=1}^{n} d_{i2}} \cdots\cdots g_l^{\sum_{i=1}^{n} d_{il}} \cdot H_2(t)^{\beta N} \bmod N^2 \\
&= g_{Ag}^{a_1 \sum_{i=1}^{n} d_{i1}} \cdot g_{Ag}^{a_2 \sum_{i=1}^{n} d_{i2}} \cdots\cdots g_{Ag}^{a_l \sum_{i=1}^{n} d_{il}} \cdot H_2(t)^{\beta N} \bmod N^2 \\
&= g_{Ag}^{a_1 \sum_{i=1}^{n} d_{i1} + a_2 \sum_{i=1}^{n} d_{i2} + \cdots + a_l \sum_{i=1}^{n} d_{il}} \cdot H_2(t)^{\beta N} \bmod N^2
\end{aligned} \tag{2}
$$

### 3.5 Fault Tolerance Handling

As mentioned in Section 2.1, if some users' smart meters don't work well, smart meters might take a break from reporting or reset it later. Here we use $\hat{\mathbb{U}} \subset \mathbb{U}$ to represent the users with broken smart meters and $\mathbb{U} / \hat{\mathbb{U}}$ to represent the rest users with normal smart meters. Then these scenarios will greatly influence the character of the blinding factors such that $\sum_{i \in \mathbb{U}/\hat{\mathbb{U}}} \pi_i \neq 0 \bmod N$. This will directly lead to get the wrong aggregation results for the aggregator. Depending on such occasions, we now modify some parts of our scheme to make the aggregation go well.

The initialization phase and the report phase will not change. During the beginning part of the aggregation phase, once the aggregator discovers that the total number of the users' reports is not $n$, it then sends the set $\hat{\mathbb{U}}$ to TA. After receiving the set $\hat{\mathbb{U}}$, TA calculates

$\hat{H} = H_2(t)^{\sum_{i\in\tilde{U}}\pi_i}$ and sends $\hat{H}$ back to the aggregator. Then the aggregator can continue calculate $V' = \hat{H} \cdot H_2(t)^{\pi_0} \prod_{i\in U/\tilde{U}} CT_i \bmod N^2$ as (3).

$$V' = \left(g_{Ag}\right)^{a_1\sum_{i\in U/\tilde{U}} d_{i1} + a_2\sum_{i\in U/\tilde{U}} d_{i2} + \cdots + a_l\sum_{i\in U/\tilde{U}} d_{il}} \left(H_2(t)\right)^{\beta N} \bmod N^2 \tag{3}$$

Here we take $M' = a_1 \sum_{i\in U/\tilde{U}} d_{i1} + a_2 \sum_{i\in U/\tilde{U}} d_{i2} + \cdots + a_l \sum_{i\in U/\tilde{U}} d_{il}$ and $R = \left(H_2(t)\right)^{\beta}$. And we can see the report $V' = g_{Ag}{}^{M'} R^N \bmod N^2$ is still a ciphertext of the Paillier Cryptosystem. Thus the aggregator can get $M'$ by taking the decryption algorithm of the Paillier Cryptosystem. By taking the Algorithm 1, the aggregator can recover the summation of the data with the same type $\left(D_1', D_2', \ldots, D_l'\right)$ where each $D_j' = \sum_{i\in U/\tilde{U}} d_{ij}$.

## 4. Extensions

### 4.1 Extension to Support Time-of-Use Electricity Pricing Mode

Time-of-use electricity pricing mode divides one day into several different time periods and sets up corresponding electricity price for these time periods. In each time period, smart meter collects the total power consumption and then multiplies by the corresponding price to get the electricity fees for the user. So when the aggregator needs to obtain a fixed user's electricity fees in a certain time period, our scheme can support such demand.

Assume one day is divided into $T_1, T_2, \ldots, T_m$ and the corresponding price is $\beta_1, \beta_2, \ldots, \beta_m$. The collection cycle of the aggregator covers $\omega$ time periods, i.e. from $T_1$ to $T_\omega$, from $T_{\omega+1}$ to $T_{2\omega}$, and so on. We use $\Sigma_{i1}, \Sigma_{i2}, \ldots, \Sigma_{im}$ to represent the total electricity usages of a fixed user $U_i$ in each time period. We take the first cycle as an example. It is to say that we will show how the aggregator computes the electricity fees for user $U_i$ during $T_1$ to $T_\omega$. In report phase, for $j = 1,2,\ldots,\omega-1$, $U_i$ computes the ciphertext as $CT_{i,j} = g^{\beta_j \Sigma_{ij}} H_2\left(T_j\right)^{\pi_i} \bmod N^2$ and for the last one $T_\omega$, the ciphertext is modified as $CT_{i,\omega} = g^{\beta_\omega \Sigma_{i\omega}} \hat{H}_\omega^{\pi_i} \cdot r_i^N \bmod N^2$, where $r_i \in Z_N^*$ is a random number and $\hat{H}_\omega$ is constructed as (4).

$$\hat{H}_\omega = \left(\prod_{j=1}^{\omega-1} H_2\left(T_j\right)\right)^{-1} \bmod N^2 \tag{4}$$

In the aggregation phase, the aggregator collects $\omega$ ciphertexts and aggregates them as following (5).

$$\begin{aligned} CT_i &= CT_{i,\omega} \cdot \prod_{j=1}^{\omega-1} CT_{i,j} \bmod N^2 \\ &= g^{\beta_1 \Sigma_{i1} + \beta_2 \Sigma_{i2} + \cdots + \beta_\omega \Sigma_{i\omega}} \cdot \left( \prod_{j=1}^{\omega-1} H_2(T_j) \right)^{\pi_i} \cdot \left( \hat{H}_\omega \right)^{\pi_i} \cdot r_i^N \bmod N^2 \\ &= g^{\sum_{j=1}^{\omega} \beta_j \Sigma_{ij}} \cdot r_i^N \bmod N^2 \end{aligned} \tag{5}$$

Therefore the aggregator can get $\sum_{j=1}^{\omega} \beta_j \Sigma_{ij}$ by taking decryption algorithm of the Paillier Cryptosystem. And $\sum_{j=1}^{\omega} \beta_j \Sigma_{ij}$ is the electricity fees for user $U_i$ during $T_1$ to $T_\omega$.

### 4.2 Extension to Support Dynamic Users

In this section, we will discuss the case that some users may join or leave the system at any time period. So we need to modify some parts of the proposed scheme so that the aggregation can still be successful no matter users' addition or removal.

Here we assume that at any time period, the added users set is $\mathrm{U}_a$ and the removed users set is $\mathrm{U}_b$. So in the initialization phase, TA first generates blinding factors $\{\pi_i\}_{i \in \mathrm{U}_a}$ for new users. Then TA gives the aggregator $\pi_0' = \pi_0 - \sum_{a \in \mathrm{U}_a} \pi_a + \sum_{b \in \mathrm{U}_b} \pi_b$ as the new blinding factor. Here we take $\tilde{\mathrm{U}} = \mathrm{U} + \mathrm{U}_a - \mathrm{U}_b$ as the new updated users. In the report phase, the aggregator will receive a set of reports $\{\sigma_i, CT_i\}_{i \in \tilde{\mathrm{U}}}$. Since the batch verification won't be influenced, so if they are valid for all the signatures, the aggregator computes the value $V = H_2(t)^{\pi_0'} \prod_{i \in \tilde{\mathrm{U}}} CT_i \bmod N^2$ as (6).

$$\begin{aligned} V &= g_{Ag}^{a_1 \sum_{i \in \tilde{\mathrm{U}}} d_{i1} + a_2 \sum_{i \in \tilde{\mathrm{U}}} d_{i2} + \cdots + a_l \sum_{i \in \tilde{\mathrm{U}}} d_{il}} H_2(t)^{\pi_0' + \sum_{j \in \tilde{\mathrm{U}}} \pi_j} \bmod N^2 \\ &= g_{Ag}^{a_1 \sum_{i \in \tilde{\mathrm{U}}} d_{i1} + a_2 \sum_{i \in \tilde{\mathrm{U}}} d_{i2} + \cdots + a_l \sum_{i \in \tilde{\mathrm{U}}} d_{il}} H_2(t)^{\pi_0' + \sum_{i=1}^{n} \pi_i + \sum_{j \in \mathrm{U}_a} \pi_a - \sum_{j \in \mathrm{U}_b} \pi_b} \bmod N^2 \\ &= g_{Ag}^{a_1 \sum_{i \in \tilde{\mathrm{U}}} d_{i1} + a_2 \sum_{i \in \tilde{\mathrm{U}}} d_{i2} + \cdots + a_l \sum_{i \in \tilde{\mathrm{U}}} d_{il}} H_2(t)^{\sum_{i=0}^{n} \pi_i} \bmod N^2 \\ &= g_{Ag}^{a_1 \sum_{i \in \tilde{\mathrm{U}}} d_{i1} + a_2 \sum_{i \in \tilde{\mathrm{U}}} d_{i2} + \cdots + a_l \sum_{i \in \tilde{\mathrm{U}}} d_{il}} H_2(t)^{\beta N} \bmod N^2 \end{aligned} \tag{6}$$

Then the aggregator takes the decryption algorithm of the Paillier Cryptosystem and Algorithm 1 to get ($D_1$, $D_2$,…, $D_l$) where $D_j = \sum_{i \in \tilde{\mathrm{U}}} d_{ij}$. We can see that when users join the system, TA only needs to generate blinding factors for the new users and change the blinding factor for the aggregator. And when users leave the system, we only need TA to generate new blinding factor for the aggregator. All the other users do not need to change anything for aggregation. So it's obvious that this scheme can handle the dynamic user scenarios in an efficient way.

## 5. Security and Performance Analysis

### 5.1 Security Analysis

**Against External Attackers:** The communication flows from users to the aggregator can be

eavesdropped by external attackers but they cannot get users' electricity consumption information from the encrypted data. That is to say, although external attackers can get the data $CT_i = g_1^{d_{i1}} \cdot g_2^{d_{i2}} \cdot \cdots \cdot g_l^{d_{il}} \cdot \left(H_2(t)\right)^{\pi_i} \bmod N^2$ from user $U_i$, they can't get the information about ($d_{i1}$, $d_{i2}$,…, $d_{il}$) because $\pi_i$ is unknown to attackers. So external attackers can't get the value $\left(H_2(t)\right)^{\pi_i}$. And they can't know ($d_{i1}$, $d_{i2}$, …, $d_{il}$). Therefore, the electricity consumption information about users is secure against the external attackers.

**Against Internal Attackers:** As we mentioned before in the system model, the aggregator is untrusted. So here we'd like to refer the aggregator as the internal attacker. We can see that the aggregator obtains all the users' encrypted data $\{CT_i\}_{i=1}^n$ and the value ($D_1$, $D_2$,…, $D_l$). First the aggregator can't get each user's electricity usage data because the value $\pi_i$ is unknown to the aggregator. So the aggregator can't know the value $\left(H_2(t)\right)^{\pi_i}$ which means the aggregator won't know ($d_{i1}$, $d_{i2}$,…, $d_{il}$). Second, although the aggregator gets the aggregated result $D_j = \sum_{i=1}^n d_{ij}$, it still unable to obtain personal user electricity consumption data ($d_{i1}$, $d_{i2}$,…, $d_{il}$). So the electricity consumption information about users is secure against the internal attackers.

**Traceability:** If the batch verification fails, our scheme has the ability to find out which users' signatures are invalid by finding invalid signature algorithm in [18].

**Unforgeability:** The signature part of our scheme is unforgeable under the assumption of the standard CDH problem. We give the proof of unforgeability as follow.

**Proof of Unforgeability**

In random oracle model, the signature part of our scheme is existentially unforgeable under the assumption of the standard CDH problem in multiplicative cyclic groups. According to the theorem 3.2 in [21], we prove our signature is unforgeable as follows.

**Definition 1**. Order-$q$ group $\mathbb{G}_1$ is a $(t',\varepsilon')$-bilinear group if $\mathbb{G}_1$ satisfies the following properties:

- A group $\mathbb{G}_2$ of order $q$ and a bilinear map $e:\mathbb{G}_1\times\mathbb{G}_1\to\mathbb{G}_2$ exist, and $e$ is computable in time at most $t'$.
- No algorithm $\varepsilon$-breaks CDH on $\mathbb{G}_1$.

**Definition 2**. In a chosen-message attack [22], if a signature scheme $(KeyGen, Sign, Verify)$ satisfies existential unforgeability, the scheme is defined by the following game between an adversary $\mathbb{A}$ and a challenger:

**Setup.** The challenger can get a public key $PK$ and private key $SK$ by running algorithm $KeyGen$. The adversary $\mathbb{A}$ can obtain $PK$.

**Queries.** $\mathbb{A}$ can adaptively requests at most $q_s$ messages $M_1,\ldots,M_{q_s} \in \{0,1\}^*$ with $PK$. The challenger responds to each signature $\sigma_i = Sign(SK_i, M_i)$ which is queried.

**Output.** If $M$ is not in ($M_1,\ldots,M_{q_s}$) and $Verify(PK, M, \sigma) = \text{valid}$, $\mathbb{A}$ outputs a pair $(M,\sigma)$ and wins the game

$\mathsf{AdvSig}_{\mathrm{A}}$ is defined as the probability that A wins in the above game, taken over the coin tosses of *KeyGen* and of A.

**Definition 3**. If a forger A makes at most $q_s$ signature queries and $q_{H_1}$ queries for the hash function $H_1$ and $q_{H_3}$ queries for the hash function $H_3$, A is called $\left(t,q_{H_1},q_{H_3},q_s,\varepsilon\right)$-breaks a signature scheme in time at most $t$ with $\mathsf{AdvSig}_{\mathrm{A}}$ being at least $\varepsilon$. If no forger $\left(t,q_{H_1},q_{H_3},q_s,\varepsilon\right)$-breaks a signature scheme, the signature scheme is $\left(t,q_{H_1},q_{H_3},q_s,\varepsilon\right)$-existentially unforgeable under an adaptive chosen-message attack.

**Theorem 1**. Let $\mathrm{G}_1$ be a $\left(t',\varepsilon'\right)$-multiplicative cyclic group of order $q$. Under an adaptive chosen-message attack, the signature scheme proposed on $\mathrm{G}_1$ *is* $\left(t,q_{H_1},q_{H_3},q_s,\varepsilon\right)$-secure against existential forgery.

**Proof**. If a forger A can $\left(t,q_{H_1},q_{H_3},q_s,\varepsilon\right)$-breaks the signature scheme, there is a $t'$-time algorithm B to solve standard CDH on $\mathrm{G}_1$ with probability $\varepsilon'$ at least.

Algorithm B can give an instance $\left(q,g,g^a,g^b\right)$ of the CDH problem to output $g^{ab}$ with a generator $g$ of $\mathrm{G}_1$. Algorithm B simulates the challenger and interacts with forger A as follows:

**Setup**. Algorithm B starts by giving A the generator $g$ and the public key $Y_i = g^a$.

***H*-queries**. Algorithm A can query the random oracle at all times.

For $H_1$-query on $t_j$ :

1. If the query $t_j$ already appears on the $H_1$-list in a tuple ($t_j$, $H_1$-$coin_j$, $k_j$, $l_j$) then algorithm B retrieves ($k_j$, $l_j$) from $H_1$-list.
2. Otherwise, a random coin $H_1\text{-}coin_j \in \{0,1\}$ will be generated by B. If $H_1\text{-}coin_j = 1$, B generates $k_j \in Z_q^*$ and $l_j = 0$; else B generates $k_j, l_j \in Z_q^*$. B logs ($t_j$, $H_1$-$coin_j$, $k_j$, $l_j$) in the $H_1$-list.
3. B responds with $W = H_1\left(t_j\right) = g^{k_j} \cdot \left(g^b\right)^{l_j}$.

For $H_3$-query on $CT_i$:

1. If the query $CT_i$ has already appeared on the $H_3$-list in a tuple ($CT_i$, $n_i$), algorithm B retrieves $n_i$ from $H_3$-list.
2. Otherwise, B generates $n_i \in Z_q^*$ and logs ($CT_i$, $n_i$) in the $H_3$-list.
3. B responds $h_i=H_3(CT_i)=n_i$.

**Signature Queries.** While A requests a signature on ($CT_i$, $t_j$), B makes the following response to the query.

1. Algorithm B can obtain $H_1\left(t_j\right)$ by running the above algorithm for responding to $H_1$-queries. The corresponding tuple is ($t_j$, $H_1$-$coin_j$, $k_j$, $l_j$) on the $H_1$-list. When $H_1$-$coin_j$=0, let B abort.
2. Otherwise, we know $H_1$-$coin_j$=1 and hence $W = H_1\left(t_j\right) = g^{k_j}$ and $h_i=H_3(CT_i)=n_i$. Define $\sigma_i = V_i = (g^a)^{k_j n_i}$. Observe that $e\left(g,\sigma_i\right) = e\left(W,Y_i^{h_i}\right)$ and so $\sigma_i$ is a valid signature on

($CT_i$, $t_j$). Algorithm $\mathrm{A}$ can get $\sigma_i$ from algorithm $\mathrm{B}$.

**Output.** Finally, algorithm $\mathrm{A}$ products a signature $\sigma_f$ on ($CT_f$, $t_s$) and $CT_f$ is without signature query. If there is no tuple containing $CT_f$ and $t_s$ on the $H_1$-list and $H_3$-list, $\mathrm{B}$ issues a query for $H_1(t_s)$ and $H_3(CT_f)$ by itself to ensure there is such a tuple. Assume $\sigma_f$ is a valid signature on ($CT_f$, $t_s$); if not, $\mathrm{B}$ aborts. Then the tuple ($t_s$, $H_1$-*coin*, $k$, $l$) can be found by algorithm $\mathrm{B}$ on the $H_1$-list and the tuple ($CT_f$, $n$) on the $H_3$-list. When $H_1$-*coin*=1, $\mathrm{B}$ aborts. Otherwise, $\mathrm{B}$ can derive its CDH problem answer $g^{ab}$ from the following (7).

$$\begin{aligned} \sigma_f &= W^{ah} \\ &= H_1(t_s)^{aH_3(CT_f)} \\ &= \left(g^k \cdot \left(g^b\right)^l\right)^{an} \\ &= \left(g^{k+bl}\right)^{an} \\ &= g^{ank} \cdot g^{abnl} \end{aligned} \tag{7}$$

The description of algorithm $\mathrm{B}$ is completed.

**Data Integrity:** Since the signature part of our scheme is proved secure under the CDH problem in the random oracle model, data integrity can be guaranteed.

**Batch Verification Security:** Our scheme satisfies batch verification security and it can be proved in the following.

**Proof of Batch Verification**

According to the security proof of [19], first $Verify(M_1, SK_1, \sigma_1) = \cdots = Verify(M_n, SK_n, \sigma_n) = 1$ implies that $Batch((M_1, SK_1, \sigma_1), \ldots, (M_n SK_n \sigma_n)) = 1$. Let vector $\Delta = (\delta_1, \delta_2, \ldots, \delta_n)$ where each $\delta_i$ is $\ell_b$ bits random element in $Z_q^*$. The following (8) represent the verification equation.

$$\begin{aligned} e\left(\prod_{i=1}^{n} \sigma_i^{\delta_i}, g\right) &= \prod_{i=1}^{n} e\left(\sigma_i^{\delta_i}, g\right) \\ &= \prod_{i=1}^{n} e\left(H_1(t)^{SK_i H_3(M_i)}, g\right) \\ &= \prod_{i=1}^{n} e\left(H_1(t)^{H_3(M_i)}, g^{SK_i}\right) \\ &= \prod_{i=1}^{n} e\left(H_1(t)^{H_3(M_i)}, PK_i\right) \end{aligned} \tag{8}$$

Since $\sigma_i$, $H_1(t)^{H_3(M_i)}$ and $PK_i$ are all in $\mathrm{G}_1$ for all $i$, we can rewrite $\sigma_i = g^{\alpha_i}$, $H_1(t)^{H_3(M_i)} = g^{r_i}$ and $PK_i = g^{X_i}$ for all $\alpha_i, r_i, X_i \in Z_q^*$. So the verification equation can be rewritten to (9).

$$e\left(\prod_{i=1}^{n} g^{\alpha_i \delta_i}, g\right) = \prod_{i=1}^{n} e\left(g^{r_i \delta_i}, g^{X_i}\right) \Rightarrow e(g,g)^{\sum_{i=1}^{n} \delta_i \alpha_i} = e(g,g)^{\sum_{i=1}^{n} \delta_i r_i X_i} \tag{9}$$

If we set $\beta_i = \alpha_i - r_i X_i$, then we can obtain (10).

$$\begin{aligned} & e(g,g)^{\sum_{i=1}^{n} \delta_i \alpha_i - \sum_{i=1}^{n} \delta_i r_i X_i} = 1 \\ & \Rightarrow \sum_{i=1}^{n} \delta_i \alpha_i - \sum_{i=1}^{n} \delta_i r_i X_i \equiv 0 \, mod \, q \\ & \Rightarrow \sum_{i=1}^{n} \delta_i \beta_i \equiv 0 \, mod \, q \end{aligned} \tag{10}$$

If $Batch\left((M_1, SK_1, \sigma_1), \ldots, (M_n SK_n \sigma_n)\right) = 1$, but we find out that for some $j$, there exists such a situation that $Verify\left(M_j, PK_j, \sigma_j\right) = 0$. Because $q$ is a prime, so $\beta_j$ has an inverse $\gamma_j$ such that $\beta_j \gamma_j \equiv 1 mod \, q$. Thus, we can set $j = 1$ and $\delta_1 \equiv -\gamma_1 \sum_{i=2}^{n} \delta_i \beta_i \, mod \, q$. And now we can see that $Verify\left(M_1, PK_1, \sigma_1\right) = 0$ but $Batch\left((M_1, SK_1, \sigma_1), \ldots, (M_n SK_n \sigma_n)\right) = 1$. Obviously this breaks batch verification. So we define $E$ be an event that $Verify\left(M_1, PK_1, \sigma_1\right) = 0$ holds but $Batch\left((M_1, SK_1, \sigma_1), \ldots, (M_n SK_n \sigma_n)\right) = 1$. Note that we make no assumptions about the remaining values.

Let the last *n*-1 values of $\Delta$ be $\Delta' = \delta_2, \ldots, \delta_n$ and $|\Delta'|$ denotes the number of possible values for this vector. From above we know there is exactly one value of $\delta_1$ for a fixed vector $\Delta'$, which will make event $E$ happen. Given $\delta_1$ by randomly chosen, the probability of $E$ is $Pr[E \mid \Delta'] = 2^{-\ell_b}$. So, when we choose $\delta_1$ at random and sum up all possible choices of $\Delta'$, $Pr[E] \le \sum_{i=1}^{|\Delta'|} \left(Pr[E / \Delta'] \cdot Pr[\Delta']\right)$ can be obtained. When the values are plugged, we can obtain $Pr[E] \le \sum_{i=1}^{2^{\ell_b (n-1)}} \left(2^{-\ell_b} \cdot 2^{-\ell_b (n-1)}\right) = 2^{-\ell_b}$. The advantage is negligible for valid batch verification over invalid signature.

## 5.2 Performance Analysis

**Comparison.** In this part, our scheme is compared with other schemes [1, 5, 9, 13] about some security features as in **Table 1**.

**Table 1.** Comparison of Features

| | Our scheme | [1] | [5] | [9] | [13] |
|---|---|---|---|---|---|
| Against external attackers | √ | √ | √ | √ | √ |
| Against internal attackers | √ | | | √ | √ |
| Traceability | √ | | | | |
| Data integrity | √ | √ | | | √ |
| Secure batch verification | √ | | | | √ |
| Time-of-use electricity bill | √ | | | | |
| Fault tolerance | √ | | | √ | |
| Dynamic user | √ | | | √ | |
| Formal proof | √ | √ | | | √ |

**Computational Cost.** In this part, we will evaluate the computational complexity of our scheme in two aspects. One is the computational costs of each user and the aggregator throughout the whole process. The other is the computational costs about the data aggregation and batch verification. The following is a detailed description.

First, our scheme is compared with Lu et al.'s scheme [1] about the computational costs of each user and the aggregator throughout the whole process. As for our scheme, when one user generates a ciphertext $CT_i$ of his electricity usage data ($d_{i1}$, $d_{i2}$,…, $d_{il}$), it requires ($l$+1) exponentiation operations and $l$ multiplication operations. Moreover, it requires 1 exponentiation operation and 1 multiplication operation to generate the signature $\sigma_i$ for the user. Therefore, a total of $l$+2 exponentiation operations and $l$+1 multiplication operations are required for user. When the aggregator receives the total $n$ ciphertexts from users, it requires 2 pairing operations, 2$n$ exponentiation operations and 2($n$-1) multiplication operations for batch verification. And the aggregator requires 1 exponentiation operation and $n$ multiplication operations for generating the value $V$. As for decryption, it requires 1 paillier cryptosystem decryption operation [1]. Therefore, the aggregator requires a total of 2 pairing operations, 2$n$+1 exponentiation operations, 3$n$-2 multiplication operations and 1 paillier cryptosystem decryption operation. In [1], the local GW verifies the signatures and aggregates the encrypted data while OA recovers the aggregated report and gets the total aggregated data. So the computational complexity of the aggregator in our scheme will be compared with the GW and OA together in Lu et al.'s scheme [1]. According to [1], we make the comparison in **Table 2**. This paper defines the computational complexity costs of a pairing operation, an exponentiation operation, a multiplication operation and Paillier Cryptosystem Decryption by $C_p$, $C_e$, $C_m$ and $C_{pai}$ respectively.

**Table 2.** Comparison of Computational Complexity

| | Our scheme | Lu et al.[1] |
|---|---|---|
| User | $(l+2)C_e+(l+1)C_m$ | $4C_p+(l+1)C_e+C_m$ |
| Aggregator | $2C_p+(2n+1)C_e+(3n-2)C_m+C_{pai}$ | $(n+5)C_p+C_e+5C_m+C_{pai}$ |

Furthermore, **Table 3** shows the time costs of all operations according to [13].

**Table 3.** Time Costs of Operation

| Notations | Descriptions | Time Cost (ms) |
|---|---|---|
| $C_p$ | Pairing Operation | ≈ 49.25 |
| $C_e$ | Exponentiation Operation | ≈ 1.57 |
| $C_m$ | Multiplication Operation | ≈ 0.19 |
| $C_{pai}$ | Paillier Cryptosystem Decryption | ≈ 1.57 |

According to the operating costs, **Fig. 3(a)** describes the variation of each user's computational costs and **Fig. 3(b)** describes the variation of the aggregator's computational costs in terms of $l$. we can see that there is an unknown number $n$ represents the total user numbers in **Table 2**. In **Fig. 3(b)**, we simply assume $n$=100. In reality, the total users' number is larger; the advantage will be more obvious in our scheme. So it is clear that we distinctly reduce the computational complexity.

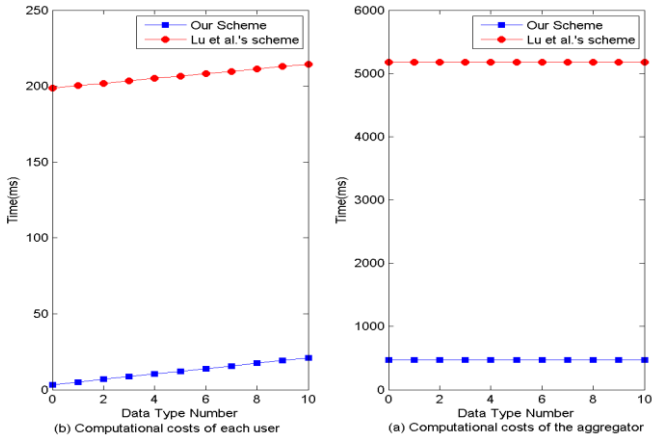



**Fig. 3.** Computational cost of each user and the aggregator

Second, we compare our scheme with the schemes in [1, 13] in the computational costs about the data aggregation and batch verification. For homogeneity, we set $l$=1 both in our scheme and in [1]. Our scheme requires 2 exponentiation operations and 1 multiplication operation to calculate $CT_i$ for each user and 1 exponentiation operation and $n$ multiplication operations for the aggregator to aggregate ciphertexts of all users. Therefore, our scheme requires $2n$+1 exponentiation operations and $2n$ multiplication operations in all. According to the schemes in [1, 13], the computational costs of aggregation are $(3n+2)C_e + 3nC_m$ and $(4n+2)C_p+(2n+2)C_e +(3n+3)C_m$ respectively. We make the comparison in **Table 4** and **Fig. 4**.

**Table 4.** Comparison of Aggregation

| Protocol | Costs |
|---|---|
| Our scheme | $(2n+1)C_e +2nC_m$ |
| Fan et al.'s scheme [13] | $(3n+2)C_e +3nC_m$ |
| Lu et al.'s scheme [1] | $(4n+2)C_p+(2n+2)C_e +(3n+3)C_m$ |

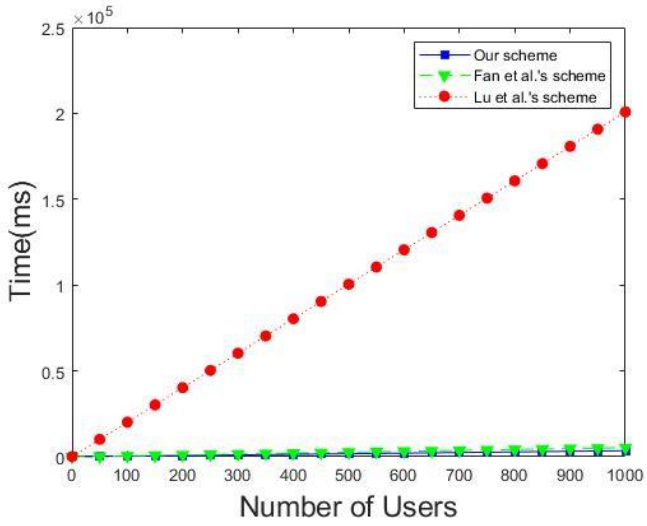



**Fig. 4.** Performance Comparison of Aggregation

As for batch verification, our scheme requires 2 pairing operations, $2n$ exponentiation operations and $2(n\text{-}1)$ multiplication operations for aggregator to batch verify all signatures. According to the schemes in [1, 13], the computational costs of batch verification are $(n+1)C_p+(2n+1)C_e+(n+1)C_m$ and $(n+1)C_p+(n+1)C_m$ respectively. We make the comparison in **Table 5** and **Fig. 5**.

**Table 5.** Comparison of Batch Verification

| Protocol | Costs |
|---|---|
| Our scheme | $2C_p+2nC_e+2(n+1)C_m$ |
| Fan et al.'s scheme [13] | $(n+1)C_p+(2n+1)C_e+(n+1)C_m$ |
| Lu et al.'s scheme [1] | $(n+1)C_p+(n+1)C_m$ |

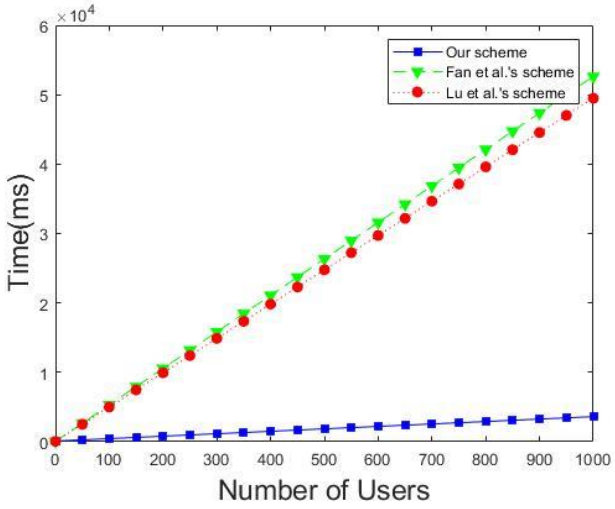



**Fig. 5.** Performance Comparison of Batch Verification

**Communication Overhead.** In this part, we will evaluate the computation overhead of user-to-aggregator in our scheme. Each user generates the encrypted data $CT_i$ and $\sigma_i$, then sends them to the aggregator. So the size should be $S_z=|CT_i|+|\sigma_i|$. If $N$ is 1024-bit and $G_1$ is 160-bit, then the size $S_z$=2048+160. So the total communication overhead is $S=n\cdot S_z$ from user to aggregator for $n$ users and $l$ types electricity usage data. For each dimensional data, each user generates a 2048-bit ciphertext in [13]. So if they have to transmit $l$ types data, the communication overhead is $S'=(2048\cdot l+160)\cdot n$ in total. We plot the communication overhead of Fan et al.'s scheme [13] in **Fig. 6(a)** and our scheme in **Fig. 6(b)** according to user number $n$ and data type $l$.

From the above performance analysis, our scheme obviously meets more security features but has less computational complexity and lower communication overhead. So our scheme is suitable to be applied into the smart grid communications.

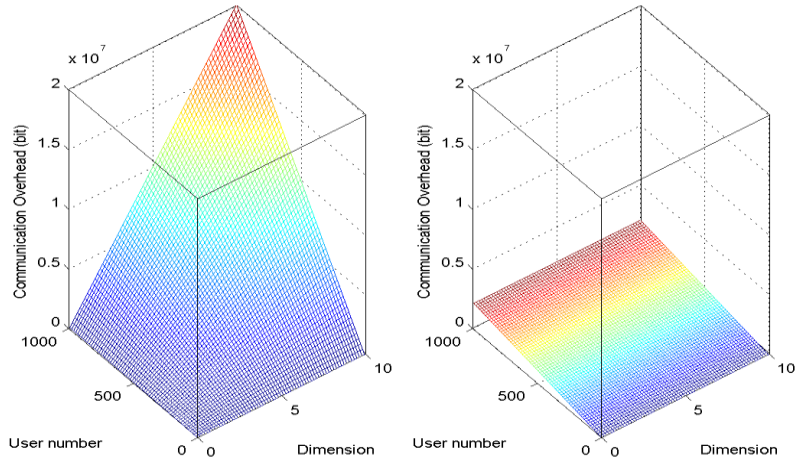



**Fig. 6.** Communication Overhead between User and Aggregator

## 6. Conclusion

In this paper, a data aggregation scheme is proposed. Our scheme has many excellent properties. First, we take the multidimensional data which is rarely mentioned in researches on smart grid into account. Second, though each user has multidimensional data, we use the Paillier Cryptosystem to encrypt the multidimensional data as a whole and take advantage of the homomorphic property to achieve data aggregation demand. Third, we apply blinding factor technique into our scheme so our scheme can resist the internal attackers on the security level. Fourth, our scheme is able to support fault tolerance so that even some smart meters don't work, the aggregation process can still work well. Fifth, we construct efficient batch verification that reduces the computational complexity from 2*n* to 2 pairing operations. Sixth, our batch verification is suitable to use the technique in [18] to find invalid signatures if the batch verification fails. Seventh, our scheme can be extended to support time-of-use electricity pricing mode and dynamic users. Eighth, we provide security analysis that our scheme can resist external attackers and internal attackers and give detailed proof of unforgeability security and batch verification security. Ninth, through performance analysis, the computational costs and communication overhead can be significantly reduced in our scheme. In the future, we will study the possible attack named human-factor-aware differential aggregation attack and extend our scheme to resist such attack.

## References

[1] R. Lu, X. Liang, X. Li, X. Lin, and X. Shen, "EPPA: An efficient and privacy-preserving aggregation scheme for secure smart grid communications," *IEEE Transations on Parallel Distributed Systems*, vol. 23, no. 9, pp. 1621-1631, Sep. 2012. Article (CrossRef Link)

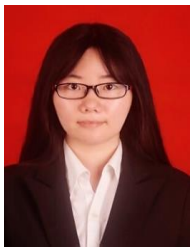
[2] R. Anderson and S. Fuloria, "Who controls the off switch?" in *Proc. of IEEE International Conference on Smart Grid Communicaions*, pp. 96-101, 2010. Article (CrossRef Link)

[3] W. Jia, H. Zhu, Z. Cao, X. Dong, and C. Xiao, "Human-factor-aware privacy-preserving aggregation in smart grid," *IEEE Systems Journal*, vol. 8, no. 2, pp. 598-607, June 2014. Article (CrossRef Link)

[4] P. Paillier, "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes," in *Proc. of Advances in Cryptology-EUROCRYPT*, vol. 1592, pp. 223-238, 1999. Article (CrossRef Link)
[5] F. Li, B. Luo, and P. Liu, "Secure information aggregation for smart grids using homomorphic encryption," in *Proc. of the 1st IEEE International Conference on Smart Grid Communications*, pp. 327-332, 2010. Article (CrossRef Link)
[6] F. D. Garcia and B. Jacobs, "Privacy-friendly energy-metering via homomorphic encryption," in *Proc. of Security and Trust Management*, pp. 226-238, 2010. Article (CrossRef Link)
[7] H. Li, X. Lin, H. Yang, X. Liang, R. Lu, and X. Shen, " EPPDR: An efficient privacy-preserving demand response scheme with adaptive key evolution in smart grid," *IEEE Transactions on Parallel Distributed Systems*, vol. 25, no. 8, pp. 2053-2064, Aug. 2014. Article (CrossRef Link)
[8] F. Borges, and M. Mühlhäuser, "EPPP4SMS: Efficient privacy-preserving protocol for smart metering systems and its simulation using real-world data," *IEEE Transactions on Smart Grid*, vol. 5, no. 6, pp. 2701-2708, Nov. 2014. Article (CrossRef Link)
[9] L. Chen, R. Lu, and Z. Cao, "PDAFT: A privacy-preserving data aggregation scheme with fault tolerance for smart grid communications," *Peer-to-Peer Networking Applications*, vol. 8, no. 6, pp. 1122-1132, Nov. 2015. Article (CrossRef Link)
[10] K. Shim and C. Park, "A secure data aggregation scheme based on appropriate cryptographic primitives in heterogeneous wireless sensor networks," *IEEE Transactions on Parallel Distributed Systems*, vol. 26, no. 8, pp. 2128-2139, Aug. 2015. Article (CrossRef Link)
[11] S. B. Othman, A. A. Bahattab, A. Trad, and H. Youssef, "Confidentiality and integrity for data aggregation in WSN using homomorphic encryption," *Wireless Personal Communications*, vol. 80, no. 2, pp. 867-889, Jan. 2015. Article (CrossRef Link)
[12] S. Verma, P. Pillai, and Y. F. Hu, "Energy-efficient privacy homomorphic encryption scheme for multi-sensor data in WSNs," in *Proc. of the 7th IEEE International Conference on Communication Systems and Networks (COMSNETS)*, pp. 1-6, 2015. Article (CrossRef Link)
[13] C. Fan, S. Huang, and Y. Lai, "Privacy-enhanced data aggregation scheme against internal attackers in smart grid," *IEEE Transactions Industrial Informatics*, vol. 10, no. 1, pp. 666-675, Feb. 2014. Article (CrossRef Link)
[14] V. Rastogi and S. Nath, "Differentially private aggregation of distributed time-series with transformation and encryption," in *Proc. of ACM SIGMOD International Conference on Management of data*, pp. 735-746, 2010. Article (CrossRef Link)
[15] E. Shi, T. H. H. Chan, E. Rieffel, R. Chow, and D. Song, "Privacy-preserving aggregation of time-series data," in *Proc. of NDSS Symposium*, 2011. Article (CrossRef Link)
[16] H. Bao and R. Lu, "A lightweight data aggregation scheme achieving privacy preservation and data integrity with differential privacy and fault tolerance," *Peer-to-Peer Networking and Applications*, vol. 10, no. 1, pp. 106-121, Sep. 2015. Article (CrossRef Link)
[17] H. Bao and R. Lu, "A new differentially private data aggregation with fault tolerance for smart grid communications," *IEEE Internet of Things Journal*, vol. 2, no. 3, pp. 248-258, June 2015. Article (CrossRef Link)
[18] L. Law and B. J. Matt, "Finding invalid signatures in paring-based batches," in *Proc. of IMA International Conference on Cryptography and Coding*, pp. 34-53, 2017. Article (CrossRef Link)
[19] J. Camenisch, S. Hohenberger, and M. Pedersen, "Batch verification of short signatures," *Journal of Cryptology*, vol. 25, no. 4, pp. 723-747, Oct. 2011. Article (CrossRef Link)
[20] M. Bellare, J. A. Garay, and T. Rabin, "Fast batch verification for modular exponentiation and digital signatures," in *Proc. of International Conference on the Theory and Applications of Cryptology-EUROCRYPT*, vol. 1403, pp. 236-250, 1998. Article (CrossRef Link)
[21] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing," *Journal of Cryptology*, vol. 17, no. 4, pp. 297-319, July 2004. Article (CrossRef Link)
[22] S. Goldwasser, S. Micali, and R. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks," *SIAM Journal Computing*, vol. 17, no. 2, pp.281-308, 1988. Article (CrossRef Link)

**Yueyu Zhang** received his MS and PhD degree from Xidian University in 2005 and 2008. He was a visiting scholar at Michigan State University from 2014 to 2015. Now he is an associate professor of Xidian University. His research interests include cryptographic protocols, security in Internet of Things and wireless network.

**Jie Chen** is an associate professor of Xidian University. She received her MS and PhD degree from Xidian University in 2005 and 2007. Her research interests include cryptographic protocols, design and analysis of cipher algorithm, security in Smart Grid.



**Hua Zhou** received her MS degree from Xidian University in 2017. Her main research interests include key management, group signature, smart grid and wireless network security.

**Lanjun Dang** received her M.E. degree and PH.D. degree in Communication and Information Systems from Xidian University, Xi'an, China, in 2005 and 2008, respectively. Now she is an associate professor of Xidian University. Her research interests included the security of mobile IP networks, authentication in wireless sensor networks, and information security.