sensors

Article

Privacy-Preserving Outsourcing Algorithms for Multidimensional Data Encryption in Smart Grids

Feng Zhai [1,2], Ting Yang [1,*], Bing Zhao [2] and Hao Chen [2]

[1] School of Electrical Engineering and Automation, Tianjin University, Weijin Road No. 92, Tianjin 300072, China; zaifeng_17@tju.edu.cn
[2] China Electric Power Research Institute, State Grid, 15 Xiaoying East Road No. 15, Beijing 300072, China; zhaob@epri.sgcc.com.cn (B.Z.); chenhao2010@epri.sgcc.com.cn (H.C.)
* Correspondence: yangting@tju.edu.cn

Abstract: With the development of the Internet of Things, smart grids have become indispensable in our daily life and can provide people with reliable electricity generation, transmission, distribution and control. Therefore, how to design a privacy-preserving data aggregation protocol has been a research hot-spot in smart grid technology. However, these proposed protocols often contain some complex cryptographic operations, which are not suitable for resource-constrained smart meter devices. In this paper, we combine data aggregation and the outsourcing of computations to design two privacy-preserving outsourcing algorithms for the modular exponentiation operations involved in the multi-dimensional data aggregation, which can allow these smart meter devices to delegate complex computation tasks to nearby servers for computing. By utilizing our proposed outsourcing algorithms, the computational overhead of resource-constrained smart meter devices can be greatly reduced in the process of data encryption and aggregation. In addition, the proposed algorithms can protect the input's privacy of smart meter devices and ensure that the smart meter devices can verify the correctness of results from the server with a very small computational cost. From three aspects, including security, verifiability and efficiency, we give a detailed analysis about our proposed algorithms. Finally, through carrying out some experiments, we prove that our algorithms can improve the efficiency of performing the data encryption and aggregation on the smart meter device side.

Keywords: smart grid; privacy preserving; data aggregation; modular exponentiation; outsourcing computation

check for updates

Citation: Zhai, F.; Yang, T.; Zhao, B.; Chen, H. Privacy-Preserving Outsourcing Algorithms for Multidimensional Data Encryption in Smart Grids. *Sensors* **2022**, *22*, 4365. https://doi.org/10.3390/s22124365

Academic Editor: Hossam A. Gabbar

Received: 16 May 2022
Accepted: 5 June 2022
Published: 9 June 2022

1. Introduction

Smart grids [1] play an important role in promoting social stability and the economic development, as they can utilize the information and communication technologies [2] to stably and efficiently generate and distribute electricity. The grid providers can formulate a more reasonable and reliable distribution strategy of electricity by the real-time collection and analysis of the situation of electric generation, transmission and demand/consumption [3]. However, the information exchange between the end user and the grid providers may face some security issues and threats [4–6].

The first issue is how to distinguish and authenticate the identity of a new smart meter (SM) or gateway (GW) when it wants to connect to the smart grid system [7,8]. Another issue is how to ensure the privacy and integrity of the collected data from a SM [9,10]. Integrity means to ensure that the exchanged data between the various parts of smart grid will not be modified or deleted. Privacy means to ensure that the collected data will not be leaked to the adversary. The electricity consumption data often contains the user's confidential information. Once this confidential information is leaked or distorted, it may cause serious harm to the consumers or the grid system [11]. For example, the adversary can analyze the electricity consumption in different time periods to determine whether the owner is at home.

Recently, a number of researchers have designed many multidimensional data encryption and aggregation protocols [12–14] to solve the above issues. Although these protocols continually strive to improve the efficiency and enhance the protection of users' data, these protocols often contain some complex cryptographic operations, such as modular exponentiation operation. In practical application, for the resource-constrained SM, there may be no insufficient computation resources and storage resources to carry out the modular exponentiation operation. The resource-constrained SM may take a long time to execute the designed protocol, which may not meet the requirements for real-time processing grid data. How to improve the execution efficiency of data encryption and aggregation has become an urgent problem to be solved.

Outsourcing computation [15,16] can allow the resource-constrained SM to outsource complex computation tasks to the server (may be another GW) with sufficient computation resources, which is suitable for SMs. However, there are three security challenges [17] for outsourcing computation. Trusted computing [18] may solve these challenges, but it needs corresponding hardware devices. Therefore, it has become more and more popular to design some secure outsourcing algorithms to solve these challenges. The specific challenges are described below. At first, the server is not completely trusted by the user. The outsourced data often contains the confidential data, such as the plaintext of the collected data and the user's private key in smart grid, which cannot be disclosed to the server. The first challenge is how to protect the privacy of the user's data in the process of outsourcing computation. Due to the various subjective factors including saving computation resources and objective factors, such as existing system bugs, attacked by hackers, the server may be curious [19,20], lazy and malicious, and thus may try to steal users' confidential data and return random or computationally indistinguishable results to cheat users. The second challenge is to ensure that the user has the capability to verify the correctness of the returned results. The third challenge is efficiency. The time cost of the secure outsourcing algorithm on the local user side must be lower than that of solving this task by itself. Otherwise, it will be meaningless for the user to execute an outsourcing algorithm. Generally speaking, an outsourcing algorithm must be privacy-preserving, verifiable and efficient.

In this paper, we combine outsourcing computation with data encryption and aggregation to reduce the computational overhead on the SM side and improve the execution efficiency of SMs. Specifically, we explore how to outsource the modular exponentiation operation involved in these data encryption and aggregation protocols to a powerful server. In the designed outsourcing algorithms, SMs can protect the confidentiality and privacy of data by utilizing random splitting technology. In addition, a SM can verify the correctness of results from an untrusted server with a high probability and low computational overhead. We can summarize the contributions of this paper into the following aspects:

- For the different forms of modular exponentiation operation in some data encryption and aggregation protocols, we respectively propose two different outsourcing algorithms. The first outsourcing algorithm can solve modular exponentiation with a fixed base and a variable exponent. The second outsourcing algorithm can solve modular exponentiation with a variable base and exponent. Both of these two outsourcing algorithms can protect the privacy of user's data by logic division and the Euler theorem;
- The two outsourcing algorithms ensure that a SM can verify the correctness of the returned results. In the first algorithm, an SM can detect a server's malicious behavior with a probability of 19/20. In the second algorithm, incorrect results from a malicious server can be detected by an SM with a probability of 59/60;
- The two outsourcing algorithms only need one round of communication between the server and SM. Through systematic analysis, we can prove that the proposed algorithms satisfy all security requirements including privacy, verifiability, and efficiency. In addition, we carry out a comprehensive experiment to demonstrate that the proposed algorithms are efficient.

We organize the rest of our paper as follows: in Section 2, we review the related work. We introduce the system model and the threat model and give some formal definitions of secure outsourcing computation in Section 3. In Section 4, we briefly describe a specific data encryption and aggregation protocol and give a detailed description of the proposed outsourcing algorithms. The security and complexity analysis are given in Section 5. In Section 6, we evaluate the proposed algorithms through experiments. After that, we conclude this paper in Section 7.

2. Related Work

In this section, we will introduce some privacy-preserving algorithms for smart grids and review some related works about securely outsourcing modular exponentiation to a malicious server. Table 1 shows the difference between our algorithms and the previous algorithms.

Table 1. Comparison of algorithms.

Algorithm	Data Type	Technique	Trusted Authority	Lightweight
Li [21]	One-dimensional	Paillier	Yes	No
Lu [22]	Multi-dimensional	Paillier	Yes	No
Boudia [14]	Multi-dimensional	Elliptic curve	Yes	No
Our	Multi-dimensional	Paillier	No	Yes

2.1. Privacy-Preserving Algorithms for Smart Gird

Saxena et al. [23] proposed a secure and efficient mutual authentication and authorization algorithm in smart grids with an advanced metering infrastructure (AMI). The authors used an attribute-based access control to mitigate outsider and insider threats in smart grids. Their algorithm was suitable for the different user-roles scenario. However, their proposed algorithm could not consider the error detection and the fault tolerance. For secure smart grid communications, Sun et al. [24] put forward an efficient aggregation algorithm (APED) with error detection by employing a pairwise private stream aggregation method. In order to improve the APED algorithm, Shi et al. [25] proposed a diverse grouping-based algorithm (DG-APED) for data aggregation with error detection. The authors applied the differential privacy technique into the grouping-based private stream aggregation and formulated the lifetime of a SM as an exponential distribution to achieve the data aggregation and perform error detection in a malfunctioning SM. In their algorithm, a control center (CC) could only obtain the aggregated results but not the individual data. Bao et al. [26] put forward a new differentially private data aggregation algorithm with fault tolerance named DPAFT. Their algorithm could handle fault tolerance in smart metering by applying a novel key management technique and constructing an artful constraint relation. Based on an improved Boneh–Goh–Nissim cryptosystem, their algorithm could protect the privacy of the user's data under the honest-but-curious model. Guo et al. [27] put forward a lightweight privacy-preserving data aggregation algorithm by utilizing a novel symmetric homomorphic encryption scheme. In addition, the authors also proposed an authentication agreement algorithm based on the password authenticated key exchange protocol. In order to satisfy the requirement of the fine-grained demands from CC, Li et al. [21] designed a privacy-preserving multisubset data aggregation algorithm named PPMA. Their algorithm could not only aggregate the user's electricity consumption data of the different ranges, but also protect the privacy of user's electricity consumption data from being disclosed to a strong adversary. Ge et al. [28] put forward a consortium blockchain-oriented approach to solve the issue of user's privacy in energy trading in smart grids. The authors designed an account mapping technique to avoid directly exposing user's data to attackers. Gough et al. [29] designed an innovative differential privacy-compliant algorithm to protect the data from SMs. Their algorithm ensured that the extra costs are divided among the participants by a fair, efficient and equitable manner based on the cooperative game theory.

2.2. Outsourcing Algorithms for Modular Exponentiation

The research on the outsourcing modular exponentiation algorithm [30–36] mainly focuses on two aspects: two servers and a single server. Under the two-servers model, the user can verify the correctness of the returned result by comparing the results returned from the two servers. Hohenberger et al. [30] gave a formal security definition for securely outsourcing cryptographic computation. The authors also proposed two practical and secure outsourcing algorithms for the Cramer–Shoup cryptosystem and the Schnorr signature. However, the probability that the user could detect malicious behavior from the servers was only 1/2. In order to improve efficiency and verifiability on the user side, Chen et al. [31] put forward new algorithms for the secure outsourcing of modular exponentiation. The verifiability in their algorithms was improved to 2/3. In addition, the authors also designed a secure outsourcing algorithm for simultaneous modular exponentiation. Ye et al. [32] explored how to securely outsource modular exponentiation to malicious servers by utilizing a new logical division method. The user could verify the validity of the returned results with the probability of 19/20. There are also a number of studies that concentrate on designing secure outsourcing algorithms for modular exponentiation under a single server. Wang et al. [33] firstly explored how to securely outsource modular exponentiation to a malicious server. However, the user needed to perform a time-consuming modular exponentiation operation locally once. The verifiability in their algorithm was only 1/2. Based on the Euler theorem, Ren et al. [34] presented an outsourcing algorithm for modular exponentiation under a single server. While improving efficiency, the user could check the failure with a probability of 1. However, their algorithm required the user to pre-compute some random tuples. Zhou et al. [35] put forward a secure outsourcing algorithm for modular exponentiation without pre-computation. The authors designed this new method to protect the privacy of the base, exponentiation and modular exponentiation. The user could detect malicious behavior of the server with a probability of approximately 1.

3. System Model and Definition

In this section, we will give a brief description about the system model and the threat model. Then, we introduce the general framework and some security requirements of the secure outsourcing computation.

3.1. System Model and Threat Model

As shown in Figure 1, the system architecture of a secure outsourcing computation model in a smart grid mainly consists of four entities: smart meters (SM), gateways (GW), control centers (CC) and a malicious server (MS). A detailed description of the functions of these entities is as follows:

- SM: The main function of SMs is to continuously collect various electricity consumption information and other identity information of users, such as the user's name and location. The SM periodically encrypts the collected data by the modular exponentiation operations and sends it to the GW;
- GW: The main function of a GW is to verify the legitimacy of the received messages and aggregate data reported by multiple SMs. Then, the GW sends the results after aggregation to the CC;
- CC: The main function of a CC is to generate system parameters. When a new device (SM or GW) is connected to the grid network, it will be authenticated by a CC. In addition, a CC can verify the legitimacy of the received messages from SMs and GWs.
- MS: The main function of a MS is to help a SM complete the corresponding complex computation tasks. The SM sends the time-consuming computation tasks to the MS, and the MS returns a correct result to the SM. The MS only communicates with the SM. There is no communication channel between the MS and the GW (or CC).

In our system architecture, we define a SM as an honest entity. The SM will carry out the corresponding operations according to protocol. However, the SM does not have sufficient computation resources and storage resources to complete complex computations. The GW and CC are honest but curious. They will execute protocol honestly, but they try to get the user's private data during the execution of the protocol. The MS is regarded to be a malicious entity with sufficient computation resources. The MS not only wants to steal the user's confidential data, but also returns computationally indistinguishable results to deceive the user. This threat model is called the "full-malicious" model [37] and is used in this paper. Therefore, the SM should encrypt the inputs x and outputs y of a computation task F before sending it to the MS and should have capability to verify the correctness of the returned results from the MS.

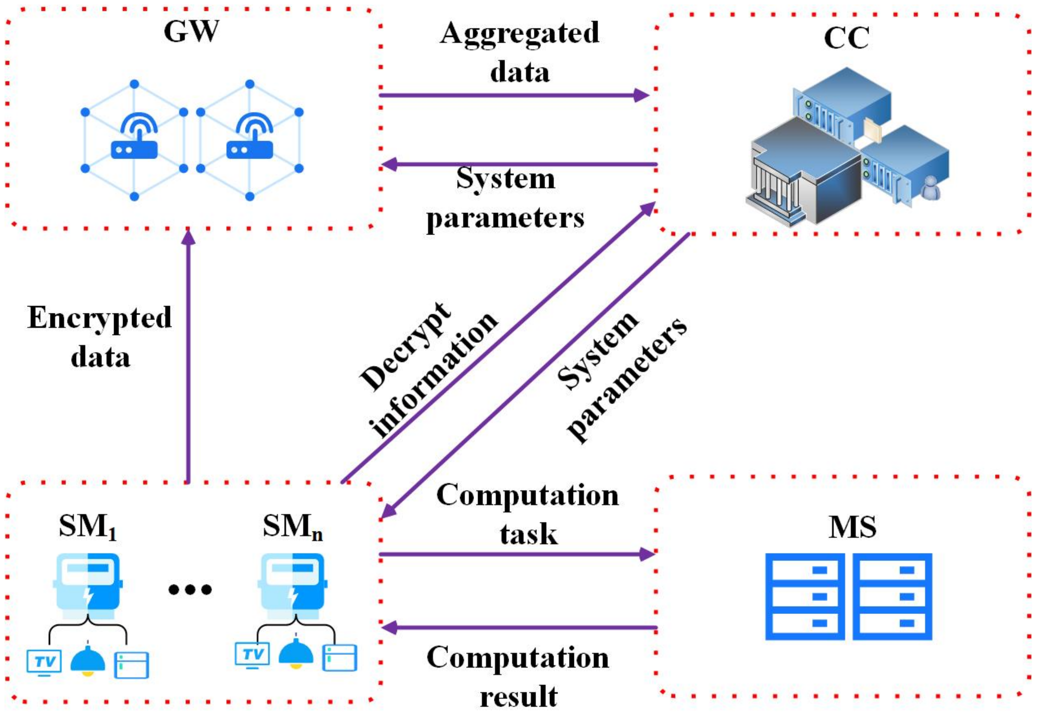


Figure 1. System model.

3.2. Rand Algorithm

The Rand algorithm [38] is used to generate a series of blinded pairs, shaped like (c, g_1^c). The specific process of the algorithm is as follows:

Z_p is a multiplicative group of order M, and g_1 is its generator. The user firstly generates n random integers $\alpha_1, \alpha_2, \ldots, \alpha_n \in Z_M$ and computes $\beta_j = g_1^{\alpha_j}$. Then, the user stores α_j and β_j in a table. When the user needs a blinded pair, the user will select k numbers from $1, 2, \ldots, n$ as S. For each $j \in S$, the user selects a random integer $z_j \in 1, 2, \ldots, h$, where $h > 1$, and computes:

$$z \equiv \sum_{j \in S} \alpha_j z_j (mod \quad M), \qquad Z \equiv \prod_{j \in S} \beta_j^{z_j} (mod \quad p).$$

The user finally obtains a blinded pair (z, Z). For a l-bit exponent, for each invoke, the computational complexity of Rand algorithm is $O(log^2\, l)$.

3.3. General Framework

The general framework of our proposed outsourcing algorithms contains the following sub-algorithms:

- **Pre-computation:** The SM computes some static parameters in advance to speed up the execution efficiency of algorithms;

- **Problem transformation:** At the stage of problem transformation, the SM encrypts the computation input x into a public value x' and stores a secret value γ locally, which can be used for decryption and verification of the results. Then, the SM sends the computation task F and the public value x' to the MS;
- **Server computation:** Based on the received computation task and the value x', the MS solves the transformed problem $F(x')$ and returns the corresponding result y' to the SM;
- **Result verification:** At the stage of result verification, the SM verifies the validity the result y' based on the stored γ;
- **Result recovery:** If the result successfully passes the verification, the SM decrypts y' and obtains the real result y.

3.4. Security Requirements

We follow the security definition introduced in [30], which has been widely employed in many previous outsourcing algorithms. A cryptographic algorithm Alg can be divided into two parts: a trusted component T and an untrusted component U. T has access to make queries to U. The adversary A also can be divided into two parts: the adversarial environment E and a malicious component U' that operates in place of U. E can submit the adversarial input to Alg and U' can record all information during the execution of Alg. We consider that (T, U) is an outsource-secure implementation of Alg if (1) T and U implement Alg, i.e., $Alg = T^U$, and (2) a malicious U' cannot learn anything about the inputs and outputs during the execution of Alg. Furthermore, we formally define the security of the outsourcing computation for the cryptographic algorithm.

Definition 1 (**Algorithm with Outsource-IO**). *An algorithm Alg obeys outsource input/output specification if it takes in five inputs $(x_{hs}, x_{hp}, x_{hu}, x_{ap}, x_{au})$ and generates three outputs (y_s, y_p, y_u). The first three inputs are generated by T and the last two inputs are adversarially submitted by E. The classification of these five inputs depends on how $A = (E, U')$ has knowledge about them. In particular,*

Inputs:

- *x_{hs} is the honest and secret input, which is only known to T;*
- *x_{hp} is the honest and protected input, which is known to T and E, but not to U';*
- *x_{hu} is the honest and unprotected input, which is known to T, E and U';*
- *x_{ap} is the adversarial and protected input, which is known to T and E, but not to U';*
- *x_{au} is the adversarial and unprotected input, which is known to T, E and U'.*

Outputs:

- *y_s is the secret output, which is only known to T;*
- *y_p is the protected output, which is known to T and E, but not to U';*
- *y_u is the unprotected output, which is known to T, E and U'.*

Definition 2 (**Outsource-Security**). *Let Alg be an algorithm with outsource-IO. A pair of algorithms (T, U) can be considered to be an outsource-secure implementation of Alg if:*

Correctness: T^U is a correct implementation of Alg.

Security: For anyone probabilistic polynomial-time adversary $A = (E, U')$, there exists the probabilistic expected polynomial-time simulators (S_1, S_2) such that the following pairs of random variables are computationally indistinguishable.

(1) **Pair One:** $EVIEW_{real} \sim EVIEW_{ideal}$.

If the random variables of **pair one** are computationally indistinguishable, this means the adversarial environment E cannot learn any useful information.

We can formally define the view obtained by the adversarial environment E in the real processing process as follows:

$$
\begin{aligned}
EVIEW^{i}_{real} = \{(IS^{i}, x^{i}_{hs}, x^{i}_{hp}, x^{i}_{hu}) \leftarrow I(1^{k}, IS^{i-1});\\
(ES^{i}, j^{i}, x^{i}_{ap}, x^{i}_{au}, stop^{i}) \leftarrow E(1^{k}, EVIEW^{i-1}_{real}, x^{i}_{hp}, x^{i}_{hu});\\
(TS^{i}, US^{i}, y^{i}_{s}, y^{i}_{p}, y^{i}_{u}) \leftarrow\\
T^{U'(US^{i-1})}(TS^{i-1}, x^{j^{i}}_{hs}, x^{j^{i}}_{hp}, x^{j^{i}}_{hu}, x^{i}_{ap}, x^{i}_{au}) :\\
(ES^{i}, y^{i}_{p}, y^{i}_{u})\}.\\
EVIEW_{real} = EVIEW^{i}_{real} \quad if \quad stop^{i} = TRUE.
\end{aligned}
$$

The real process proceeds in rounds. There is an honest and stateful process I, which the environment E does not have access to. In round i, I can pick the honest (secret, protected and unprotected) inputs.

Then, E can choose some parameter values according to its view from the last round. The value of ES^{i} is a way of remembering what it does next time when it is invoked. j^{i} is the index of inputs $(x^{i}_{hs}, x^{i}_{hp}, x^{i}_{au})$ (E can only specify the index, but cannot specify their values of these inputs). E can choose the adversarial and protected input x^{i}_{ap} and the adversarial and unprotected input x^{i}_{au}. $stop^{i}$ is the Boolean variable, which determines whether the round i is the last round in the process.

Based on the inputs $(TS^{i-1}, x^{j^{i}}_{hs}, x^{j^{i}}_{hp}, x^{j^{i}}_{hu}, x^{i}_{ap}, x^{i}_{au})$, the algorithm $T^{U'}$ begins to run, where TS^{i-1} is previously saved by T and generates a new state TS^{i} for T. In addition, $T^{U'}$ produces the secret output y^{i}_{s}, the protected output y^{i}_{p} and the unprotected output y^{i}_{u}. The oracle U' is given its previously saved state US^{i-1}, as input, and the current state of U' is saved in the variable US^{i}. E can obtain the final view in the real process, which is just its view in the last round.

$$
\begin{aligned}
EVIEW^{i}_{ideal} = \{(IS^{i}, x^{i}_{hs}, x^{i}_{hp}, x^{i}_{hu}) \leftarrow I(1^{k}, IS^{i-1});\\
(ES^{i}, j^{i}, x^{i}_{ap}, x^{i}_{au}, stop^{i}) \leftarrow E(1^{k}, EVIEW^{i-1}_{ideal}, x^{i}_{hp}, x^{i}_{hu});\\
(AS^{i}, y^{i}_{s}, y^{i}_{p}, y^{i}_{u}) \leftarrow Alg(AS^{i-1}, x^{j^{i}}_{hs}, x^{j^{i}}_{hp}, x^{j^{i}}_{hu}, x^{i}_{ap}, x^{i}_{au});\\
(SS^{i}, US^{i}, Y^{i}_{p}, Y^{i}_{u}, rep^{i}) \leftarrow\\
S_{1}^{U'(US^{i-1})}(SS^{i-1}, \ldots, x^{j^{i}}_{hp}, x^{j^{i}}_{hu}, x^{i}_{ap}, x^{i}_{au}, y^{i}_{p}, y^{i}_{u});\\
(z^{i}_{p}, z^{i}_{u}) = rep^{i}(Y^{i}_{p}, Y^{i}_{u}) + (1 - rep^{i})(y^{i}_{p}, y^{i}_{u}) :\\
(ES^{i}, z^{i}_{p}, z^{i}_{u})\}.\\
EVIEW_{ideal} = EVIEW^{i}_{ideal} \quad if \quad stop^{i} = TRUE.
\end{aligned}
$$

The ideal process also proceeds in rounds. In the ideal process, there is a stateful simulator S_1, which can obtain the view when shielded from the secret input x^{i}_{hs}. In round i, based on the non-secret outputs that Alg generates, S_1 can decide to either output the values (y^{i}_{p}, y^{i}_{u}) generated by Alg or replace them with some other values (Y^{i}_{p}, Y^{i}_{u}). Whether y^{i}_{p} will be replaced with Y^{i}_{p} depends on the value of rep^{i}, which is a bit. In doing this, S_1 is allowed to query the oracle U'; additionally, U' saves its state as in the real experiment.

(2) **Pair Two:** $UVIEW_{real} \sim UVIEW_{ideal}$.

If the random variables of **pair two** are computationally indistinguishable, this means the malicious software U' cannot learn any useful information.

The view that the untrusted software U' can obtain in the real process is the same as that described in **pair one**. $UVIEW_{real} = US^i \quad if \quad stop^i = TRUE$.

$$\begin{aligned} UVIEW^i_{ideal} = \{(IS^i, x^i_{hs}, x^i_{hp}, x^i_{hu}) \leftarrow I(1^k, IS^{i-1});\\ (ES^i, j^i, x^i_{ap}, x^i_{au}, stop^i) \leftarrow\\ E(1^k, ES^{i-1}, x^i_{hp}, x^i_{hu}, y^{i-1}_p, y^{i-1}_u);\\ (AS^i, y^i_s, y^i_p, y^i_u) \leftarrow Alg(AS^{i-1}, x^{j^i}_{hs}, x^{j^i}_{hp}, x^{j^i}_{hu}, x^i_{ap}, x^i_{au});\\ (SS^i, US^i) \leftarrow S_2^{U'(US^{i-1})}(SS^{i-1}, x^{j^i}_{hu}, x^i_{au}):\\ (US^i)\}.\\ UVIEW_{ideal} = UVIEW^i_{ideal} \quad if \quad stop^i = TRUE. \end{aligned}$$

In the ideal process, there is a stateful simulator S_2, which can only know the unprotected input (x^i_{hu}, x^i_{au}) and has the access to query U'. As before, U' may maintain its state.

Definition 3 (*α*-**Efficient [39]).** *A pair of algorithms (T, U') can be considered to be an α-efficient implementation of an algorithm Alg if (1) (T, U') is an outsource-secure implementation of Alg, and (2) for any inputs x, we suppose that the time cost for a task F to be solved locally by T is t_1 and the time cost for a task F to be solved locally by a secure outsourcing algorithm is t_2, which satisfies $\frac{t_2}{t_1} \leq \alpha$.*

Definition 4 (*β*-**Verifiable [39]).** *A pair of algorithms (T, U') can be considered to be a β-verifiable implementation of an algorithm Alg if (1) (T, U') is an outsource-secure implementation of Alg, and (2) for any inputs x, if there exists the malicious behavior from U', the probability that T can detect the error is not less than β.*

Remark 1. *In this paper, we do not adopt a method [40] to show the security of our proposed algorithms in a real environment. We only prove that our proposed algorithms meet our proposed security definition through detailed theoretical analysis. In the future research, we will explore how to prove the security in a real environment according to the method proposed in [40].*

4. Algorithm

Recently, Zuo et al. [12] put forward a privacy-preserving aggregation algorithm for multidimensional data in a smart grid without a trusted authority. Their algorithm is based on the ElGamal cryptosystem, which can support distributed decryption. This algorithm needs the SM to encrypt the collected data periodically by the ElGamal encryption in the data encryption stage. We will use Zuo's algorithm as an example to describe our algorithm.

In this section, we will briefly describe Zuo's algorithm. Then, for the modular exponentiation operation in Zuo's algorithm, we give a detailed description of our proposed secure outsourcing algorithms.

4.1. Overview of Zuo's Algorithm

In Zuo's algorithm, there are six parts in total. We assume that the number of SM is n.

1. **System Initialization**: CC firstly generates a series of system parameters $\{G, G_T, g, e, H, pk_{CC}, \overrightarrow{a}, \overrightarrow{b}, (R_1, R_2, \ldots, R_k, E)\}$, where G and G_T are the multiplicative cyclic group of a large secure prime p, g is a generator of G, e is a bilinear map: $G \times G \rightarrow G_T$, H is a one-way hash function: $\{0,1\}^* \rightarrow G$, pk_{CC} is the public key of CC, $\overrightarrow{a} = \{a_1, a_2, \ldots, a_w\}$ and $\overrightarrow{b} = \{b_1, b_2, \ldots, b_k\}$ are the superincreasing sequences, and $(R_1, R_2, \ldots, R_k, E)$ are the k range values of power consumption.
2. **Registration**: All SM$_i$ and GW register with CC. SM$_i$ chooses a random number $x_i \in Z_p$ and computes the public key $pk_i = g^{x_i}$ and the signature $\sigma_i = H(ID_i||T_i)^{x_i}$,

where ID_i is the identity of the user U_i and T_i is the current timestamp. Then, SM$_i$ sends $(ID_i, T_i, \sigma_i, pk_i)$ to the CC. The CC verifies if $e(g, \sigma_i) = e(pk_i, H(ID_i||T_i))$ holds. Once the equation is established, it means that the registration is successful. The registration progress of the GW is similar. The GW randomly chooses a number x_{GW} and computes the public key pk_{GW}.

3. **Generation of Common Public Key**: Each SM$_i$ broadcasts its public key pk_i and verifies the validity of other public keys from other SMs. Then, SM$_i$ computes the common public key PK as follows:

$$PK = \prod_{i=1}^{n} pk_i = g^{x_1+x_2+\ldots+x_n}.$$

4. **Encryption of User Data**: Each SM$_i$ collects w dimensions of power consumption $(m_{i1}, m_{i2}, \ldots, m_{iw})$. $m_i = (m_{i1} + m_{i2} + \ldots + m_{iw})$ is the total power consumption data of each user. If $m_i \in [R_j, R_{j+1})$, SM$_i$ chooses a random number $r_i \in Z_p^*$ and computes the corresponding ciphertext (C_i^a, C_i^b) based on the common public key PK and g^{b_j}, where $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, k$.

$$\begin{aligned} C_i^a &= g^{r_i}, \\ C_i^b &= g^{a_1 m_{i1} + a_2 m_{i2} + \ldots + a_w m_{iw}} \cdot g^{b_j} \cdot PK^{r_i}. \end{aligned}$$

SM$_i$ computes its own signature as follows:

$$\sigma_i = H(ID_i||C_i^a||C_i^b||T_i)^{x_i}.$$

SM$_i$ sends $\{ID_i, C_i^a, C_i^b, T_i, \sigma_i, pk_i\}$ to the GW.

5. **Data Aggregation**: The GW firstly checks the timestamp T_i and verifies the validity of n signatures. After successful verification, the GW computes the aggregated ciphertext $(C^a, C^b) = (\prod_{i=1}^{n} C_i^a, \prod_{i=1}^{n} C_i^b)$ and its own signature σ_{GW}. The GW sends $\{ID_{GW}, C^a, C^b, T_{GW}, \sigma_{GW}, pk_{GW}\}$ to the CC.

6. **Decryption of aggregation Data**: The CC firstly checks the timestamp T_{GW} and verifies the signature. Each SM$_i$ is required to provide D_i and signature σ_i^d to the CC.

$$\begin{aligned} D_i &= (C^a)^{x_i} = (\prod_{i=1}^{n} C_i^a)^{x_i}, \\ \sigma_i^d &= H(ID_i||D_i||T_i^d)^{x_i}. \end{aligned}$$

where T_i^d is the current timestamp. SM$_i$ sends $\{ID_i, D_i, T_i^d, \sigma_i^d, pk_i\}$ to the CC. The CC recovers the aggregated data.

From the above description, we can know that there are many complex and time-consuming cryptography operations in Zuo's algorithm, including the modular exponentiation operation and the bilinear pairing operation. For the GW and CC, there are sufficient computation resources and storage resources to carry out these complex operations. However, for an SM, as a kind of resource-constrained device, there are not enough computation resources and storage resources to deal with the time-consuming modular exponentiation operation, or it needs a lot of time to complete these operations. In Zuo's algorithm, there are three parts involving modular exponentiation operations: registration, encryption and decryption. In the registration process, each SM only performs a modular exponentiation operation. Even if it takes a longer time, it has little effect on the efficiency of the protocol. We mainly solve the modular exponentiation operation in encryption and decryption because the SM constantly performs modular exponentiation. This situation is not suitable for the CC to process power consumption data in real time. Therefore, we design secure outsourcing algorithms to improve the efficiency of the SM's execution of modular exponentiation.

4.2. Description of Outsourcing Algorithms

In Zuo's algorithm, every time the SM sends power consumption data, the SM needs to perform the modular exponentiation operation five times. The SM needs to compute C_i^a, C_i^b, σ_i, D_i and σ_i^d. When calculating C_i^a and C_i^b, the base of modular exponentiation is fixed and secret, and the exponent is variable and secret. When calculating σ_i, D_i and σ_i^d, the base and exponent of modular exponentiation are variable and secret. For these two different cases of modular exponentiation operations, we design two different secure outsourcing algorithms.

We use u to represent the base of modular exponentiation and a to represent the exponent of modular exponentiation. p is the modulus. The SM sends u, a and p to the MS, and the MS returns $u^a \mod p$ to the SM. In the process of computing C_i^a and C_i^b, g is the generator of a group and PK is the common public key. g and PK are the fixed parameters. When designing an outsourcing algorithm for C_i^a and C_i^b, the SM needs to protect the privacy of u, a and p.

In Algorithm 1, u and p are secret and fixed, and a is secret and variable. Because u is fixed, the SM can compute two blinding pairs, u^{x_1} and u^{x_2}, in advance. In order to keep the information of a private from the MS, the SM can use x_1 and x_2 to blind the exponent a. Based on the discrete logarithm problem, the MS cannot obtain the base u from u^{x_1} and u^{x_2}. Based on the big integer factorization problem, the MS cannot obtain the base modular p from sp. In order to ensure the verifiability of the returned results, the SM needs to randomly choose a number r. The SM not only needs to send the blinding original computation u^a to the MS, but also needs to send the blinding-related computation u^{ra} to the MS. When it is necessary to verify the correctness of the result returned from the MS, the SM only needs to verify whether the equation $(u^a)^r = u^{ra}$ holds. When the verification is successful, the SM can use the locally stored p to recovery the real result u^a.

Now, we utilize an example to illustrate our proposed Algorithm 1.

Example 1. *The SM wants to compute $u^a \mod p$, where $p = 11$ is a prime, $u = 2$ and $a = 2$. The SM can compute as follows:*

1. *Pre-computation: The SM chooses $x_1 = 3$ and $x_2 = 7$. Then, The SM computes $u^{x_1} = 8$ and $u^{x_2} = 7$.*
2. *Problem transformation: The SM chooses $r = 2$ and $s = 3$. Then, the SM computes $ra = 4$ and $sp = 33$. The SM computes $t_1 = 4$ and $t_2 = 2$.*
3. *Server computation: The MS computes $c_1 = 4$ and $c_2 = 16$.*
4. *Result verification: The SM can verify the following equation:*

$$4^2 \mod 11 = 16 \mod 11.$$

5. *Result recovery: The SM can recover the real result $u^a = 4 \mod 11 = 4$.*

In process of computing σ_i, D_i and σ_i^d, the exponent x_i is the user's private key, and the base is the message summary $H()$ or the aggregated ciphertext C^a, which cannot be disclosed to MS. Once leaked, it will have a serious impact on the security of the entire algorithm. Therefore, when designing an outsourcing algorithm for σ_i, D_i and σ_i^d, the SM needs to protect the privacy of the base and the exponent, that is, the privacy of u and a.

Before introducing Algorithm 2, we firstly introduce a theorem. Let N be a positive integer, and let u be an integer which satisfies $gcd(u, N) = 1$; then we have $u^{\phi(N)} \equiv 1 \mod N$.

Theorem 1. *For $N = p_1 p_2 \dots p_m$, where $p_1, p_2, \dots, p_m$ are distinct prime numbers, we have*

$$u^{a+\gamma\phi(N)} = u^a (\mod N)$$

where $\phi(\cdot)$ is the Euler's function and γ is a random integer.

```
Algorithm 1 Secure Outsourcing of Modular Exponentiation with Public Base and Secret Exponent (Fixed Base)
Input:
    $u \in Z_p^*, a \in Z_p^*$.
Output:
    $u^a \mod p$.
  1. Pre-computation:
    • The SM randomly chooses $x_1$ and $x_2$ and pre-computes $u^{x_1}$ and $u^{x_2}$, where $gcd(x_1, p-1) = 1$ and $gcd(x_2, p-1) = 1$.
  2. Problem transformation:
    • The SM randomly chooses a number $r$ from the set $[2, 11]$ and computes $ra$.
    • The SM randomly chooses a larger prime $s$ and computes $sp$.
    • The SM computes $t_1$ and $t_2$ as:
          $t_1 = a/x_1 \mod p-1,$
          $t_2 = ra/x_2 \mod p-1.$
    • The SM sends $sp$, $u^{x_1}$, $u^{x_2}$, $t_1$ and $t_2$ to MS.
  3. Server computation:
    • The MS computes $c_1$ and $c_2$ as:
          $c_1 = (u^{x_1})^{t_1} \mod sp,$
          $c_2 = (u^{x_2})^{t_2} \mod sp.$
    • The MS returns $c_1$ and $c_2$ to the user.
  4. Result verification:
    • The SM verifies the following equation:
          $(c_1)^r \mod p = c_2 \mod p.$
  5. Result recovery:
    • The SM can recover the real result as:
          $u^a = c_1 \mod p.$
```

In Algorithm 2, $N = p$ and $\phi(N) = p - 1$. u and a are secret and variable. In order to keep the information of u private from the MS, the SM firstly runs the Rand algorithm to generate four blinding pairs and compute two blinding pairs to blind the base u. The original base u can be transformed into $ug_1^{x_1}$. Because the MS does not know any knowledge about $g_1^{x_1}$, the MS cannot obtain the base u from $ug_1^{x_1}$. By utilizing ***Theorem 1***, the SM can protect the information of a from being disclosed to the MS.The original exponent a can be transformed into $t_3 = a + k_1(p-1)$, where k_1 is a random integer. Similar to Algorithm 1, the SM chooses a random integer r and compares the two returned results to ensure the verifiability of the results. When the verification is successful, the SM can use the locally stored $g_1^{x_3}$ and p to recover the real result u^a.

Now, we utilize an example to illustrate our proposed Algorithm 2.

Example 2. *The SM wants to compute* $u^a \mod p$*, where* $p = 11$ *is a prime,* $g_1 = 2$*,* $u = 3$ *and* $a = 2$*. The SM can compute as follows:*

1. *Pre-computation: The SM chooses* $x_1 = 1$*,* $x_2 = 3$*,* $x_3 = 2$*,* $x_4 = 4$*,* $x_5 = 7$*,* $x_6 = 5$*. Then, SM computes* $g_1^{x_1} = 2$*,* $g_1^{x_2} = 8$*,* $g_1^{x_3} = 4$*,* $g_1^{x_4} = 5$*,* $g_1^{x_5} = 7$*,* $g_1^{x_6} = 10$*.*
2. *Problem transformation: The SM chooses* $r = 2$*,* $s = 3$ $k_1 = 4$ *and* $k_2 = 5$*. Then, the SM computes* $ra = 4$ *and* $sp = 33$*. The SM computes* $t_1 = 2$*,* $t_2 = 7$*,* $t_3 = 42$ *and* $t_4 = 54$*.*
3. *Server computation: The MS computes* $c_1 = 3$*,* $c_2 = 31$*,* $c_3 = 3$ *and* $c_4 = 6$*.*

4. *Result verification: The SM can verify the following equation:*

$$372^2 \mod 11 = 180 \mod 11.$$

5. *Result recovery: The SM can recover the real result* $u^a = 372 \mod 11 = 9$.

Algorithm 2 Secure Outsourcing of Modular Exponentiation with Secret Base and Exponent (Variable Base)

```
Input:
  $u \in Z_p^*, a \in Z_p^*$.
Output:
  $u^a \mod p$.
  1. Pre-computation:
    • The SM runs the Rand algorithm to generate four blinding pairs $(x_1, g_1^{x_1})$, $(x_3, g_1^{x_3})$,
      $(x_4, g_1^{x_4})$, $(x_6, g_1^{x_6})$ and randomly chooses $x_2$ and $x_5$ and computes $(x_2, g_1^{x_2})$ $(x_5, g_1^{x_5})$,
      where $gcd(x_2, p-1) = 1$ and $gcd(x_5, p-1) = 1$.
  2. Problem transformation:
    • The SM randomly chooses two numbers $k_1$ and $k_2$, and chooses a random number $r$
      from $[2, 11]$. The SM computes $ra$.
    • The SM randomly chooses a larger prime $s$ and computes $sp$.
    • The SM computes $t_1$ and $t_2$ as:
          $t_1 = (-x_3 - x_1 a)/x_2 \mod p-1,$
          $t_2 = (-x_6 - x_4 ra)/x_5 \mod p-1.$
    • The SM computes $t_3$ and $t_4$ as:
          $t_3 = a + k_1(p-1),$
          $t_4 = ra + k_2(p-1).$
    • The SM sends $(u \cdot g_1^{x_1}, t_3)$, $(g_1^{x_2}, t_1)$, $(u \cdot g_1^{x_4}, t_4)$, $(g_1^{x_5}, t_2)$ and $sp$ to the MS.
  3. Server computation:
    • The MS computes $c_1, c_2, c_3$ and $c_4$ as:
          $c_1 = (u \cdot g_1^{x_1})^{t_3} \mod sp,$
          $c_2 = (g_1^{x_2})^{t_1} \mod sp,$
          $c_3 = (u \cdot g_1^{x_4})^{t_4} \mod sp,$
          $c_4 = (g_1^{x_5})^{t_2} \mod sp.$
    • The MS sends $c_1, c_2, c_3$ and $c_4$ to SM.
  4. Result verification:
    • The SM verifies the following equation:
          $(c_1 c_2 g_1^{x_3})^r \mod p = c_3 c_4 g_1^{x_6} \mod p.$
  5. Result recovery:
    • The SM can recover the real result as:
          $u^a = c_1 c_2 g_1^{x_3} \mod p.$
```

5. Security and Complexity Analysis

In this section, we firstly analyze the correctness and the security of the proposed algorithms. Then, we give a detailed description about the verifiability of the returned results. Finally, we analyze the computational complexity of the proposed algorithms.

5.1. Security Analysis

Theorem 2. *In the malicious model, the algorithm (T, U) is an outsource-secure implementation of Algorithms 1 and 2, where the input (u, a, p) may be honest, secret; or honest, protected; or adversarial, protected.*

Proof. The correctness of these two algorithms is obvious and straightforward. We mainly focus on security. $A = (E, U')$ is a PPT adversary that interacts with a PPT algorithm T in the malicious model. We need to prove that **pair one** and **pair two** are computationally indistinguishable. □

Pair One $EVIEW_{real} \sim EVIEW_{ideal}$: If the input (u, a, p) is anything other than honest and secret, the same way that the simulator S_1 behaves as in the real execution. Under an honest and secret input (u, a, p), S_1 will behave as follows: once the information is received in the ith round, S_1 neglects it and correspondingly makes two random queries of the form $(\alpha_i, \beta_i, \theta_i)$ to U'. S_1 randomly tests one output from the program (i.e., $\beta_i^{\alpha_i} \mod \theta_i$). Once it detects an error, S_1 saves the states of itself and U', and outputs $Y_p^i = "error", Y_u^i = \emptyset, rep^i = 1$. If no error is detected, S_1 outputs $Y_p^i = \emptyset, Y_u^i = \emptyset, rep^i = 0$; otherwise, S_1 randomly chooses an element R and outputs $Y_p^i = R, Y_u^i = \emptyset, rep^i = 0$. In either case, S_1 also stores the appropriate states. In the real and ideal experiment, the input distributions to U' are computationally indistinguishable. The inputs in the ideal experiment are chosen uniformly at random. Each part of all two queries that T makes is re-randomized and computationally indistinguishable. If U' behaves honestly in the ith round, we have $EVIEW_{real}^i \sim EVIEW_{ideal}^i$. If U' behaves dishonestly in the ith round, both T and S_1 detect malicious behavior with a high probability and output *"error"*. In the real experiment, the two outputs generated by U' are blinded by a random value. Therefore, we have $EVIEW_{real} \sim EVIEW_{ideal}$. In summary, we can get $EVIEW_{real} \sim EVIEW_{ideal}$ no matter whether U' is honest or malicious.

Pair Two $UVIEW_{real} \sim UVIEW_{ideal}$: The way that the simulator S_2 behaves is as follows: Once the information is received in the ith round, S_2 neglects it and correspondingly makes two random queries of the form $(\alpha_i, \beta_i, \theta_i)$ to U'. Then, S_2 saves the states of itself and U'. These real and ideal experiments can be easily distinguished by E because the outputs in the ideal experiment are never corrupted. Because there is no channel between E and U', E cannot transmit any messages to U'. In the ith round of the real experiment, T re-randomizes its inputs to U'. S_2 generates the random and independent queries to U' in the ideal experiment. We can know $UVIEW_{real}^i \sim UVIEW_{ideal}^i$. In summary, we can get $UVIEW_{real} \sim UVIEW_{ideal}$.

5.2. Verifiability Analysis

Theorem 3. *In the malicious model, the proposed Algorithm 1 is a 19/20-verifiable secure outsourcing algorithm and the proposed Algorithm 2 is a 59/60-verifiable secure outsourcing algorithm.*

Proof. At first, we prove that Algorithm 1 is a 19/20-verifiable secure outsourcing algorithm. If the MS wants to cheat the SM, the MS may make some guesses about the value c_1, c_2 and r. In order to verify the correctness of the results, the SM needs to check the equation:

$$(c_1)^r \mod p = c_2 \mod p.$$

In order to cheat the SM, the MS needs to randomly choose a integer C and constructs the following equation:

$$(Cc_1)^r \mod p = C^r c_2 \mod p.$$

In order to make the above equation hold, the MS needs to guess the value of r and distinguish c_1 from c_1 and c_2. Because the random number r is chosen from the set $[2, 11]$, the probability that the MS can accurately guess the value of r is $1/10$. The probability that the MS can accurately distinguish c_1 from c_1 and c_2 is $1/2$. Therefore, the probability that the MS can cheat the SM is 1/20. Algorithm 1 is a 19/20 (0.95)-verifiable secure outsourcing algorithm. □

Then, we prove that Algorithm 2 is a 59/60-verifiable secure outsourcing algorithm. Similar to Algorithm 1, in order to cheat the SM, the MS needs to randomly choose a integer C and construct the following equation:

$$(Cc_1c_2g_1^{x_3})^r \mod p = C^r c_3 c_4 g_1^{x_6} \mod p.$$

On the one hand, the MS needs to guess the value of r. On the other hand, the MS needs to distinguish c_1 and c_2 from c_1, c_2, c_3 and c_4 and separately compute Cc_1c_2 and $C^r c_3 c_4$. The probability that the MS can accurately guess the value of r is 1/10. The probability that the MS can accurately distinguish c_1 and c_2 from c_1, c_2, c_3 and c_4 is 1/6. Therefore, the probability that the MS can cheat the SM is 1/60. Algorithm 2 is a 59/60 (0.983)-verifiable secure outsourcing algorithm.

Then, we compare the security, verifiability and efficiency of Algorithm 2 with some previous algorithms. As shown in Table 2, there is the comparison of security, verifiability and efficiency for the user in these outsourcing algorithms. In order for all outsourced algorithms to have the same level of security, we set $c = r = 4$ and $k = l = 29$. All random numbers x, t_1 and t_2 are larger than 2^{64}. From Table 2, we can find that Algorithm 2 is superior to [33,41] in both efficiency and security. Compared to [42], although Algorithm 2 does less MM, it needs to do one more $MInv$ and one $Rand$. However, Algorithm 2 has higher verifiability. In addition, compared with the previous three algorithms, our algorithm has higher security that can protect the modulus of modular exponentiation.

Table 2. Comparison of algorithms.

	Wang [33]	**Ye [41]**	**Kiraz [42]**	**Algorithm 2**
MM	$12 + 1.5\log x > 108$	$1.5\log r + 1.5(\log t_1 + \log t_2) + 15 > 210$	$l + k + 8\log c + 38 = 112$	$15 + 1.5\log r$
$MInv$	4	6	1	2
$Rand$	6	6	5	6
Verifiability	0.5	0.991	0.917	0.983
Modular Privacy	No	No	No	Yes

5.3. Complexity Analysis

Theorem 4. *In the malicious model, the proposed Algorithm 1 is a ((1.5log r + 3)*MM + 2*MInv)/1.5l*MM-efficient secure outsourcing algorithm, and the proposed Algorithm 2 is a ((1.5log r + 15)*MM + 2*MInv)/1.5l*MM-efficient secure outsourcing algorithm.*

Proof. We denote MM as a once modular multiplication operation and $MInv$ as a once modular inverse operation. For a l-bit exponent, the SM needs to be 1.5l times MM to compute $u^a \mod p$ by the square-and-multiply method. The bit length of u, a and p is l. □

In Algorithm 1, the process of problem transformation needs three times MM (we omit ra) and twice $MInv$. The process of result verification needs 1.5log r times MM. We omit other operations such as modular additions. Thus, the proposed Algorithm 1 is a ((1.5log r + 3)*MM + 2*$MInv$)/1.5l*MM-efficient secure outsourcing algorithm.

In Algorithm 2, the process of problem transformation needs nine times MM and twice $MInv$. The process of result verification and recovery needs 1.5log $r + 6$ times MM. Thus, the proposed Algorithm 2 is a ((1.5log r + 15)*MM + 2*$MInv$)/1.5l*MM-efficient secure outsourcing algorithm.

6. Evaluation

6.1. Numeric Analysis

In this section, we will give an analysis of communication overhead and storage space overhead.

At first, we give an analysis of communication overhead. The bit length of sp is L. In Algorithm 1, the SM needs to send sp, u^{x_1}, u^{x_2}, t_1 and t_2 to the MS. The size of these parameters is $4l + L$ bits. The MS needs to return c_1 and c_2 to the SM. The size of these parameters is $2L$ bits. To sum up, the communication overhead of Algorithm 1 is $4l + 3L$ bits. Similar to Algorithm 1, in Algorithm 2, the SM needs to send $(u \cdot g_1^{x_1}, t_3)$, $(g_1^{x_2}, t_1)$, $(u \cdot g_1^{x_4}, t_4)$, $(g_1^{x_5}, t_2)$ and sp to the MS. The size of these parameters is $8l + L$ bits. The MS needs to return c_1, c_2, c_3 and c_4 to the SM. The size of these parameters is $4L$ bits. To sum up, the communication overhead of Algorithm 2 is $8l + 5L$ bits.

Then, we give a analysis of storage space overhead. The storage space that the SM requires contains two parts: an online phase and offline phase. In the offline phase of Algorithm 1, the SM needs to store the two pre-computed pairs and the parameters $\{u, p\}$, which needs $6l$ bits storage space. In the online phase of Algorithm 1, the SM firstly needs l bit of storage space to store a. Then, the SM needs $4l + L$ bits storage space during the problem transformation stage, which contains the parameters $\{ra, s, sp, t_1, t_2\}$. We assume that s and p have the same bit length l and omit the bit length of r. The client needs $l + 2L$ bits storage space during the result verification and recovery, which contains the parameters $\{c_1, c_2, u^a\}$. To sum up, the storage space overhead of Algorithm 1 is $12l + 3L$ bits.

We can analyze Algorithm 2 in a similar way. The SM needs to $13l$ bits storage space to store six blinding pairs and p in the offline phase of Algorithm 2. In the online phase of Algorithm 2, the SM firstly needs $2l$ bits storage space to store a and u. Then, the SM needs $8l + L$ bits storage space during the problem transformation stage, which contains the parameters $\{ra, s, sp, t_1, t_2, t_3, t_4, ug_1^{x_1}, ug_1^{x_4}\}$. We omit the bit length of r, k_1 and k_2. The SM needs $l + 4L$ bits of storage space during the result verification and recovery, which contains the parameters $\{c_1, c_2, c_3, c_4, u^a\}$. To sum up, the storage space overhead of Algorithm 2 is $24l + 5L$ bits.

Table 3 shows the communication overhead and storage space overhead.

Table 3. Communication overhead and storage space overhead.

	Communication Overhead	Offline	Online	Storage Space Overhead
Algorithm 1	$4l + 3L$	$6l$	$6l + 3L$	$12l + 3L$
Algorithm 2	$8l + 5L$	$13l$	$11l + 5L$	$24l + 5L$

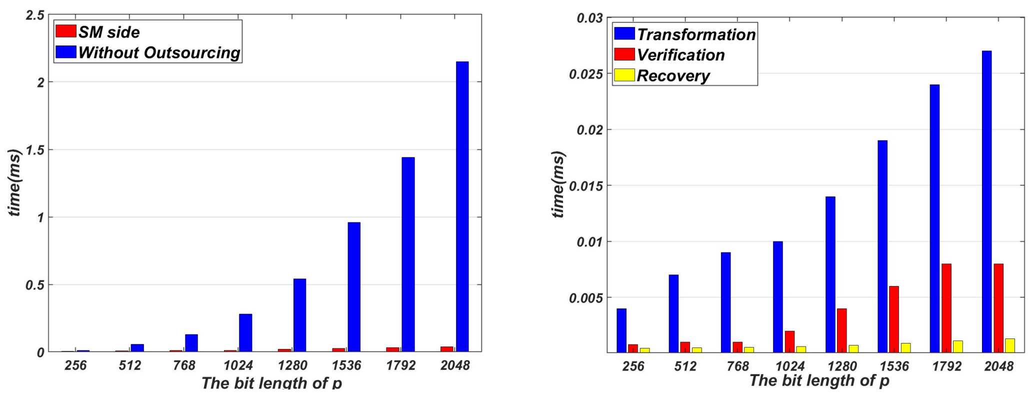
6.2. Performance Evaluation

In this section, in order to show that our proposed algorithms are efficient, we carried out some experiment evaluations. We implemented our algorithms by using the C++ programming language with the GMP library, which is specially designed to handle some large integer operations. We used a software named Sublime Text3 to write the programs. In the program, all variables were first defined as the types defined in GMP. Then, we wrote code step-by-step according to the algorithms. We need edto define many variables to receive the intermediate data. We usee the system's own time function to compute the time cost of the two algorithms. SM was simulated by a computer with a Linux Ubuntu 20.04.2 LTS operating system and Intel Core i5 processors (2.4 GMz and 2 G memory). MS was simulated by a computer with a Linux Ubuntu 20.04.2 LTS operating system and Intel Core i5 processors (2.6 GMz and 8 G memory). In our experiment, the bit length of p ranged from 256 bits to 2048 bits. As shown in Table 4, there are some simulation parameters when the bit length of p is 256 bits and 512 bits.

Table 4. Some simulation parameters.

Bit Length	*u*	*a*	*s*	*p*
256	5979749570663628312108786874212080563712524424228337985262878420394720897670892 71	77831065593910628582374740114662102962147586147297001093903417824734058418115822 47	10864091204398926481456366616012575017778580812273845319359934521304955239068542 937	82697904907611022215644205306412518781670584274273708291040583120118723036268882 41
512	754387404285988120826742806315691191408511373747561289674705476743950894451122497539359176804470461882150746408475691612284449585981855155202044474843 8435	750582098634835192302082996469078517856339381182246653167329120531958754792548491295132292354230821822750054928738939141913001392109008976698721435666 8809	861478175873681674678195073537115352012738288372116893103864237840006599576717793500595767387060711833362282500403261748910719236579350879254393697362 30279	783792629876206141918443009719853359141243678662806276651199939579569239026298451825126212286804347962362175060120125548509155767187344027224581929964 0811

Figure 2a compares the time cost of not outsourcing with our proposed Algorithm 1 on the SM side. Figure 3a compares the time cost of not outsourcing with our proposed Algorithm 2 on the SM side. As shown in Figures 2a and 3a, the time cost of Algorithms 1 and 2 are much smaller than that of direct computation. Note that the time cost of Algorithms 1 and 2 on the SM side dose not include the pre-computation process. The pre-computation process can be done off-line. In Algorithms 1 and 2, the time cost on the SM side concentrates on these three processes: transformation, verification and recovery.

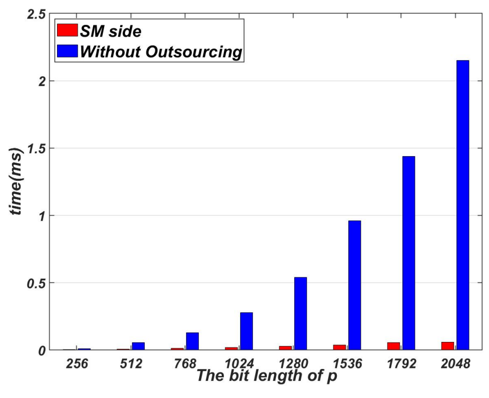


(**a**) (**b**)

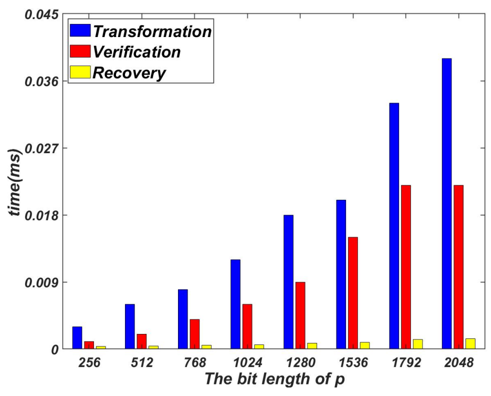
Figure 2. Evaluation results for Algorithm 1. (**a**) The time cost of Algorithm 1 without outsourcing on the SM side; (**b**) The time cost comparison among phases in Algorithm 1.

Figures 2b and 3b show the time cost of transformation, verification and recovery in Algorithms 1 and 2. As shown in Figures 2b and 3b, the process of transformation needs to take more time than two other processes. This is because the process of transformation contains two modular inverse operations. Generally speaking, it takes more time to complete a modular inversion operation than a modular multiplication operation. Because only one modular operation is performed in the process of recovery, the time cost of recovery is the smallest.

Figure 4 shows that the ratio between the time cost of our proposed algorithms and direct computation. From Figure 4, we can see that Algorithms 1 and 2 can significantly improve the efficiency of the SM. In addition, as the bit length of *p* increases, the improvement becomes more and more significant.



(**a**)



(**b**)

Figure 3. Evaluation results for Algorithm 2. (**a**) The time cost of Algorithm 2 without outsourcing on the SM side; (**b**) The time cost comparison among phases in Algorithm 2.

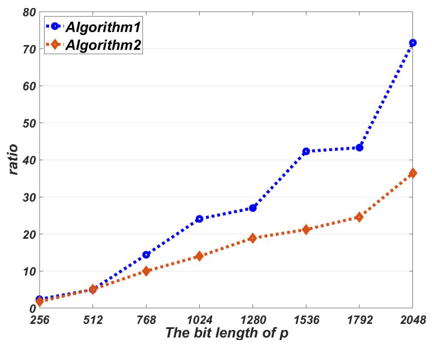


Figure 4. The ratio between the time cost of our proposed algorithms and direct computation.

7. Conclusions

In this paper, for the complex modular exponentiation operations involved in Zuo's privacy-preserving data aggregation protocol, we designed two secure and efficient outsourcing algorithms for resource-constrained SMs. The proposed algorithms not only can protect SMs' confidential data from being leaked to an untrusted server, but can also ensure the correctness of the returned results from the server. In addition, we provided an analysis of the security, verifiability and efficiency and proved that the SM can detect error with a probability of 19/20 in Algorithm 1 and with a probability of 59/60 in Algorithm 2. Finally, through experimental evaluation, we proved that our proposed algorithms are well suitable for data encryption and aggregation in smart grids. In the future, on the one hand, we will investigate more outsourcing algorithms, including bilinear pairing and scalar multiplication on elliptic curves, which are applicable to other data encryption and aggregation protocols. On the other hand, we will explore how to combine outsourcing computation with distributed computation to reduce the computational overhead on the server side.

Author Contributions: Conceptualization, F.Z. and T.Y.; methodology, F.Z. and T.Y.; software, F.Z. and B.Z.; validation, F.Z., T.Y., B.Z. and H.C.; formal analysis, F.Z.; investigation, F.Z.; resources, F.Z.; data curation, F.Z.; writing—original draft preparation, F.Z.; writing—review and editing, F.Z.; visualization, F.Z.; supervision, F.Z.; project administration, F.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Key Research and Development Program of China (2017YFE0132100) and the National Natural Science Foundation of China (61971305).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Fang, X.; Misra, S.; Xue, G.; Yang, D. Smart Grid—The New and Improved Power Grid: A Survey. *IEEE Commun. Surv. Tutor.* **2012**, *14*, 944–980. [CrossRef]
2. Gungor, V.C.; Sahin, D.; Kocak, T.; Ergut, S.; Buccella, C.; Cecati, C.; Hancke, G.P. Smart Grid Technologies: Communication Technologies and Standards. *IEEE Trans. Ind. Inform.* **2011**, *7*, 529–539. [CrossRef]
3. Yu, L.; Jiang, T.; Zou, Y. Distributed Online Energy Management for Data Centers and Electric Vehicles in Smart Grid. *IEEE Internet Things J.* **2016**, *3*, 1373–1384. [CrossRef]
4. Iyer, S. Cyber security for smart grid, cryptography, and privacy. *Int. J. Digit. Multimed. Broadcast.* **2011**, *2011*, 1–8. [CrossRef]
5. Dwivedi, A.D.; Singh, R.; Ghosh, U.; Mukkamala, R.R.; Tolba, A.; Said, O. Privacy preserving authentication system based on non-interactive zero knowledge proof suitable for Internet of Things. *J. Ambient. Intell. Humaniz. Comput.* **2021**, 1–11. [CrossRef]
6. Almaiah, M.A.; Hajjej, F.; Ali, A.; Pasha, M.F.; Almomani, O. A Novel Hybrid Trustworthy Decentralized Authentication and Data Preservation Model for Digital Healthcare IoT Based CPS. *Sensors* **2022**, *22*, 1448. [CrossRef]
7. Gope, P.; Sikdar, B. Privacy-Aware Authenticated Key Agreement Scheme for Secure Smart Grid Communication. *IEEE Trans. Smart Grid* **2019**, *10*, 3953–3962. [CrossRef]
8. Wazid, M.; Das, A.K.; Kumar, N.; Rodrigues, J.J.P.C. Secure Three-Factor User Authentication Scheme for Renewable-Energy-Based Smart Grid Environment. *IEEE Trans. Ind. Inform.* **2017**, *13*, 3144–3153. [CrossRef]
9. Ding, Y.; Wang, B.; Wang, Y.; Zhang, K.; Wang, H. Secure Metering Data Aggregation With Batch Verification in Industrial Smart Grid. *IEEE Trans. Ind. Inform.* **2020**, *16*, 6607–6616. [CrossRef]
10. He, D.; Kumar, N.; Zeadally, S.; Vinel, A.; Yang, L.T. Efficient and Privacy-Preserving Data Aggregation Scheme for Smart Grid Against Internal Adversaries. *IEEE Trans. Smart Grid* **2017**, *8*, 2411–2419. [CrossRef]
11. Yan, Y.; Qian, Y.; Sharif, H.; Tipper, D. A Survey on Cyber Security for Smart Grid Communications. *IEEE Commun. Surv. Tutor.* **2012**, *14*, 998–1010. [CrossRef]
12. Zuo, X.; Li, L.; Peng, H.; Luo, S.; Yang, Y. Privacy-Preserving Multidimensional Data Aggregation Scheme Without Trusted Authority in Smart Grid. *IEEE Syst. J.* **2020**, *15*, 395–406. [CrossRef]
13. Peng, C.; Luo, M.; Wang, H.; Khan, M.K.; He, D. An Efficient Privacy-Preserving Aggregation Scheme for Multidimensional Data in IoT. *IEEE Internet Things J.* **2021**, *9*, 589–600. [CrossRef]
14. Boudia, O.R.M.; Senouci, S.M.; Feham, M. Elliptic curve-based secure multidimensional aggregation for smart grid communications. *IEEE Sens. J.* **2017**, *17*, 7750–7757. [CrossRef]
15. Atallah, M.J.; Frikken, K.B. Securely Outsourcing Linear Algebra Computations. In Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, ASIACCS '10, Beijing, China, 13–16 April 2010; pp. 48–59.
16. Li, H.; Yu, J.; Yang, M.; Kong, F. Secure Outsourcing of Large-Scale Convex Optimization Problem in Internet of Things. *IEEE Internet Things J.* **2022**, *9*, 8737–8748. [CrossRef]
17. Ren, K.; Wang, C.; Wang, Q. Security Challenges for the Public Cloud. *IEEE Internet Comput.* **2012**, *16*, 69–73. [CrossRef]
18. Munoz, A.; Mafia, A. Software and hardware certification techniques in a combined certification model. In Proceedings of the 2014 11th International Conference on Security and Cryptography (SECRYPT), Vienna, Austria, 28–30 August 2014; pp. 1–6.
19. Gao, X.; Yu, J.; Chang, Y.; Wang, H.; Fan, J. Checking Only When It Is Necessary: Enabling Integrity Auditing Based on the Keyword with Sensitive Information Privacy for Encrypted Cloud Data. *IEEE Trans. Dependable Secur. Comput.* **2021**, 1. [CrossRef]
20. Ge, X.; Yu, J.; Zhang, H.; Bai, J.; Fan, J.; Xiong, N.N. SPPS: A Search Pattern Privacy System for Approximate Shortest Distance Query of Encrypted Graphs in IIoT. *IEEE Trans. Syst. Man, Cybern. Syst.* **2022**, *52*, 136–150. [CrossRef]
21. Li, S.; Xue, K.; Yang, Q.; Hong, P. PPMA: Privacy-preserving multisubset data aggregation in smart grid. *IEEE Trans. Ind. Inform.* **2017**, *14*, 462–471. [CrossRef]
22. Lu, R.; Liang, X.; Li, X.; Lin, X.; Shen, X. EPPA: An efficient and privacy-preserving aggregation scheme for secure smart grid communications. *IEEE Trans. Parallel Distrib. Syst.* **2012**, *23*, 1621–1631.
23. Saxena, N.; Choi, B.; Lu, R. Authentication and Authorization Scheme for Various User Roles and Devices in Smart Grid. *IEEE Trans. Inf. Forensics Secur.* **2016**, *11*, 907–921. [CrossRef]
24. Sun, R.; Shi, Z.; Lu, R.; Lu, M.; Shen, X.S. APED: An efficient aggregation protocol with error detection for smart grid communications. In Proceedings of the 2013 IEEE Global Communications Conference (GLOBECOM), Atlanta, GA, USA, 9–13 December 2013; pp. 432–437.
25. Shi, Z.; Sun, R.; Lu, R.; Chen, L.; Chen, J.; Shen, X.S. Diverse Grouping-Based Aggregation Protocol With Error Detection for Smart Grid Communications. *IEEE Trans. Smart Grid* **2015**, *6*, 2856–2868. [CrossRef]
26. Bao, H.; Lu, R. A New Differentially Private Data Aggregation With Fault Tolerance for Smart Grid Communications. *IEEE Internet Things J.* **2015**, *2*, 248–258. [CrossRef]
27. Guo, C.; Jiang, X.; Choo, K.; Tang, X.; Zhang, J. Lightweight privacy preserving data aggregation with batch verification for smart grid. *Future Gener. Comput. Syst.* **2020**, *112*, 512–523. [CrossRef]
28. Gai, K.; Wu, Y.; Zhu, L.; Qiu, M.; Shen, M.S. Privacy-Preserving Energy Trading Using Consortium Blockchain in Smart Grid. *IEEE Trans. Ind. Inform.* **2019**, *15*, 3548–3558. [CrossRef]
29. Gough, M.; Santos, S.; Alskaif, T.; Javadi, M.; Castro, R.; Catalao, J.P.S. Preserving Privacy of Smart Meter Data in a Smart Grid Environment. *IEEE Trans. Ind. Inform.* **2022**, *18*, 707–718. [CrossRef]
30. Hohenberger, S.; Lysyanskaya, A. How to Securely Outsource Cryptographic Computations. In Proceedings of the International Conference on Theory of Cryptography, Cambridge, MA, USA, 10–12 February 2005; pp. 264–282.

31. Chen, X.; Li, J.; Ma, J.; Tang, Q.; Lou, W. New Algorithms for Secure Outsourcing of Modular Exponentiations. *IEEE Trans. Parallel Distrib. Syst.* **2014**, *25*, 2386–2396. [CrossRef]
32. Ye, J.; Chen, X.; Ma, J. An Improved Algorithm for Secure Outsourcing of Modular Exponentiations. In Proceedings of the 2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops, Gwangiu, Korea, 24–27 March 2015; pp. 73–76.
33. Wang, Y.; Wu, Q.; Wong, D.S.; Qin, B.; Chow, S.S.M.; Liu, Z.; Tan, X. Securely Outsourcing Exponentiations with Single Untrusted Program for Cloud Storage. In Proceedings of the European Symposium on Research in Computer Security, Wroclaw, Poland, 7–11 September 2014; Springer: Berlin/Heidelberg, Germany, 2014; pp. 326–343.
34. Ren, Y.; Dong, M.; Qian, Z.; Zhang, X.; Feng, G. Efficient Algorithm for Secure Outsourcing of Modular Exponentiation with Single Server. *IEEE Trans. Cloud Comput.* **2021**, *9*, 145–154. [CrossRef]
35. Zhou, K.; H. Afifi, M.; Ren, J. ExpSOS: Secure and Verifiable Outsourcing of Exponentiation Operations for Mobile Cloud Computing. *IEEE Trans. Inf. Forensics Secur.* **2017**, *12*, 2518–2531. [CrossRef]
36. Li, H.; Yu, J.; Zhang, H.; Yang, M.; Wang, H. Privacy-Preserving and Distributed Algorithms for Modular Exponentiation in IoT With Edge Computing Assistance. *IEEE Internet Things J.* **2020**, *7*, 8769–8779. [CrossRef]
37. Chen, X.; Huang, X.; Li, J.; Ma, J.; Lou, W.; Wong, D.S. New Algorithms for Secure Outsourcing of Large-Scale Systems of Linear Equations. *IEEE Trans. Inf. Forensics Secur.* **2015**, *10*, 69–78. [CrossRef]
38. Nguyen, P.Q.; Shparlinski, I.E.; Stern, J. Distribution of Modular Sums and the Security of the Server Aided Exponentiation. In *Cryptography and Computational Number Theory*; Birkhäuser: Basel, Switzerland, 2001; pp. 331–342.
39. Gennaro, R.; Gentry, C.; Parno, B. Non-interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers. In *Advances in Cryptology—CRYPTO 2010*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 465–482.
40. Muñoz, A.; Maña, A.; Serrano, D. AVISPA in the validation of ambient intelligence scenarios. In Proceedings of the 2009 International Conference on Availability, Reliability and Security, Jukuoka, Japan, 16–19 March 2009; pp. 420–426.
41. Ye, J.; Wang, J. Secure outsourcing of modular exponentiation with single untrusted server. In Proceedings of the 2015 18th International Conference on Network-Based Information Systems,Taipei, Taiwan, 2–4 September 2015; pp. 643–645.
42. Kiraz, M.S.; Uzunkol, O. Efficient and verifiable algorithms for secure outsourcing of cryptographic computations. *Int. J. Inf. Secur.* **2016**, *15*, 519–537. [CrossRef]