Blockchain Based Secure Data Aggregation and Distributed Power Dispatching for Microgrids

Xinyi Luo, *Graduate Student Member, IEEE*, Kaiping Xue, *Senior Member, IEEE*, Jie Xu, Qibin Sun, *Fellow, IEEE*, and Yongdong Zhang, *Senior Member, IEEE*

***Abstract*—Power generation systems tend to be distributed and decentralized, and therefore the concept of microgrid has been proposed, which needs to implement decentralized data storage and power dispatching. The traditional power system architecture is no longer suitable in the decentralized microgrid system because there are no trusted third parties such as control centers. Therefore, it is challenging to securely implement data aggregation and power dispatching in microgrids without any trusted third party. In this paper, by leveraging blockchain, we propose secure data aggregation based on homomorphic encryption and the PBFT (Practical Byzantine Fault Tolerance) consensus, and meanwhile we propose automatic power dispatching by utilizing the PSO (Particle Swarm Optimization) algorithm and smart contracts. The security and performance analysis shows the effectiveness and efficiency of our proposed solutions.**

***Index Terms*—Blockchain, microgrid, smart contract, data aggregation, privacy preserving, power dispatching.**

I. Introduction

In THE traditional electric system, there is usually a control center to complete all power management processes [1] and to decide the plan of power generation, transmission, and distribution. However, in recent years, due to the surge in electricity consumption and the promotion of new energy power generation, traditional power systems, which are specifically designed for the centralized electric system, have begun to look beyond their capabilities [2]. We can see that the geographical span of new energy power stations has caused a heavy communication burden, and the increasing nodes have led to significant computing overhead [3]. Besides, the control center's reliability, flexibility, and security are vulnerable, and data sharing is difficult due to either lack of means or unwillingness.

Manuscript received December 23, 2020; revised April 16, 2021 and July 8, 2021; accepted July 15, 2021. Date of publication July 26, 2021; date of current version October 21, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 61972371, Grant U19B2023, and Grant U19B2044; and in part by the Youth Innovation Promotion Association of Chinese Academy of Sciences (CAS) under Grant Y202093. Paper no. TSG-01903-2020. *(Corresponding author: Kaiping Xue.)*

Xinyi Luo, Kaiping Xue, and Qibin Sun are with the School of Cyber Security, University of Science and Technology of China, Hefei 230027, Anhui, China (e-mail: kpxue@ustc.edu.cn).

Jie Xu is with the Department of Computer Science, City University of Hong Kong, Hong Kong.

Yongdong Zhang is with the Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei 230027, Anhui, China, and also with the Beijing Research Institute, University of Science and Technology of China, Beijing 100193, China.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TSG.2021.3099347.

Digital Object Identifier 10.1109/TSG.2021.3099347

Due to the centralized structure's deficiencies, the decentralized control architecture has been introduced into the power system, then the concept of the microgrid was proposed [4]. A microgrid is a small power system consists of some power generations and power consumers within a given area, and is controlled by a power management system [5]. It can operate as an independent power system or accept management from the main grid as its subnets. The decentralized microgrid system solved the problems in the centralized structure, but it brought new difficulties to power management.

For running a power management process, there are two main phases: data aggregation and power dispatching. The data aggregation phase collects the aggregation result of user power data within a particular region, and then as the input of the power dispatching algorithm to calculate the optimal power generation and distribution plan. In the existing data aggregation solutions in the electric system, e.g., [6], [7], either the aggregation gateway is assumed to be reliable, or a trusted data center is involved. However, for the microgrid system, since a single microgrid is small in scale and may be dynamically established, changed, or withdrawn, it is difficult to provide adequate secure aggregation gateways or maintain a trusted data center. Thus, existing data aggregation schemes are difficult to provide sufficient security in the microgrid system. Meanwhile, due to the difficulty of maintaining a credible authority center for each microgrid, the traditional power dispatching scheme completed by a control center is no longer feasible [8]. Therefore, in the current microgrid system, how to securely complete the data aggregation and power dispatching processes in the small-scale, distributed, and weak-trusted microgrid system becomes an urgent issue. Moreover, considering the requirements for frequent interactions between different microgrids or between microgrids and the main grid, it is necessary to establish a credible public database for data sharing between different entities, however, which is also a big challenge for traditional data center-driven power systems.

By leveraging the blockchain technology, a solid trust relationship in distributed environments can be established based on its consensus mechanisms and distributed networks. Therefore, as participants in the microgrid, including data aggregators and dispatchers, do not trust each other, blockchain can be introduced to establish a trust relationship and assist in the energy management process without a trusted third party. Therefore, in this paper, based on the blockchain technology, we provide appropriate solutions to address the challenges in microgrids. Firstly, in the data aggregation phase, it is necessary to ensure that the usage data for individual users are not leaked [9], and the aggregated results have not been tampered with, thus protecting the authenticity and correctness of the aggregated data [10]. We utilize homomorphic encryption and the PBFT (Practical Byzantine Fault Tolerance) consensus algorithm to solve the problem. Secondly, microgrids are supposed to obtain fine-grained real-time power usage data for more accurate dispatching [11]. Thus, a distributed dispatching algorithm that can securely and automatically run is needed, and we implement it based on the PSO (Particle Swarm Optimization) algorithm, smart contracts, and the above aggregation scheme. Thirdly, a public database for dispatching and auditing is required. The relevant data should be stored publicly so as to enable distributed dispatching and auditing, and we leverage blockchain technology to solve this problem naturally.

In summary, we make the following contributions.

1) By leveraging Paillier homomorphic encryption and PBFT consensus, we propose a decentralized microgrid data aggregation scheme without any authority center and resistant malicious aggregators, enabling user privacy-preserving, data integrity protecting, and public storage and sharing.
2) We propose an automatic and distributed microgrid power dispatching solution based on the PSO algorithm and Ethereum smart contracts, and the above aggregation method. Moreover, we distribute the PSO algorithm to use multiple smart contracts to complete one dispatching, therefore adapting to smart contracts' computation limitation.
3) We experimentally implement our proposed scheme with Ethereum smart contracts, verify its effectiveness and test the performance, and give a security analysis. The results show that the proposed solution well meets the security and performance requirements of microgrids.

The rest of this paper is organized as follows. Section II presents the related work. Section III introduces the background. Section IV describes the system model, security assumption and design goal. Section V expound our proposed solution. Sections VI and VII give the security and performance analysis. Section VIII concludes the paper.

II. Related Work

For user privacy preserving and data validity verification in smart grid, secure data aggregation is proposed to aggregate the user data and only send the total result to the data center so that the actual value from a particular user can be concealed. A common method is to use homomorphic encryption algorithm. Li *et al.* [6] proposed a distributed in-network aggregation approach for smart grids to aggregate data along a spanning tree. Lu *et al.* [7] proposed EPPA to efficiently aggregate multidimensional data using a superincreasing sequence distributed by a trusted operation authority. PPMA proposed by Li *et al.* [12] further improves the EPPA to support multi-subset data aggregation and security under malicious gateways. Xue *et al.* [13] also proposed an efficient and robust data aggregation scheme without a trusted authority for the smart grid, which not only ensures user's privacy and efficiency but also supports flexible dynamic user management with no need of involving a trusted authority. However, these existing work cannot provide a comprehensive solution to simultaneously implement user privacy protection, data integrity verification, public sharing, and permanent data storage, and most importantly, without trusted third parties. There are also some data aggregation schemes based on blockchain and homomorphic encryption. Ghadamyari and Samet [14] and Zheng *et al.* [15] combine blockchain and homomorphic encryption for data aggregation, though, their scenarios do not involve the issue of malicious aggregation gateways, which is a key issue in microgrid data aggregation.

For microgrid dispatching, there are different optimization goals and algorithms. Most of the existing studies aim to propose better models and dispatching schemes. Bagherian and Tafreshi [16] utilized the PSO (Particle Swarm Optimization) algorithm for microgrid power dispatching to maximize the profit of the management system. Kakigano *et al.* [17] combined fuzzy control with gain-scheduling techniques for voltage control. Khorsandi *et al.* [18] proposed a distributed control method employing the conventional droop control method, which enables accurate current sharing and desirable voltage regulation. Che *et al.* [5] proposed a three-level hierarchical coordinate strategy for power exchanges among neighboring microgrids. Although these works have conducted sufficient research on dispatching algorithms, how to securely implement these schemes in a distributed environment lacking trust is still an urgent issue to be solved. Some studies introduce blockchain to solve the problem, e.g., [19]–[22]. However, most of these work aim at building a blockchain-based distributed energy trading platform, rather than providing a microgrid management system. Although [22] uses smart contracts for microgrid control, it does not implement the complex power dispatching process, but uses smart contracts to select a subset of power sources that participate in voltage regulation.

III. Preliminaries

A. Blockchain and Smart Contract

After bitcoin was first proposed by Nakamoto [23], its underlying technology, blockchain, has gained more and more dramatic attention. Blockchain is essentially a kind of distributed database that consists of blocks in linear arrangement and is identified by height. Each block stores some data and contains the hash value of the previous block. Providing the hash value makes it difficult to tamper with any data on the blockchain because one should change every block behind the target to make the hash value right. Additionally, blockchain provides a consensus mechanism to prevent one node from continuously adding blocks, thus bring more difficulties to tamper with data because one node cannot be

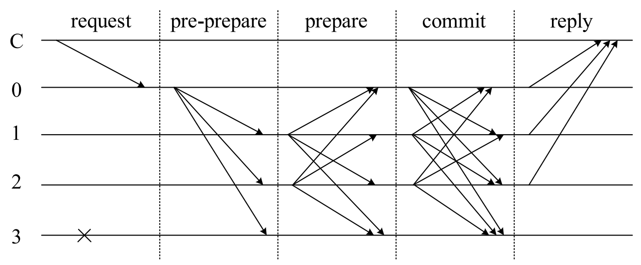

Fig. 1. The PBFT consensus algorithm.

sufficiently rapid to change all the blocks. The success of bitcoin has practically verified the security of the blockchain. The well-designed data structure and the consensus mechanism make the blockchain difficult to attack, thus provide trusts in distributed environments. Buterin *et al.* [24] further found the convergence of blockchain and smart contracts and proposed the Ethereum blockchain. Ethereum also provides an object-oriented, high-level language for implementing smart contracts - Solidity [25]. Other blockchains supporting smart contracts, such as Hyperledger, have also been developed after Ethereum. With blockchain-enabled smart contracts, it is possible to execute programs automatically with enough security and reliability, thus enable automatic dispatching in the microgrid.

B. PBFT Consensus

The PBFT consensus algorithm is proposed to solve the Byzantine Fault Tolerance (BFT) problem in distributed systems. A Byzantine fault refers to a condition of a distributed network where some nodes may fail due to malicious attacks or software errors and exhibit arbitrary behavior. The Practical Byzantine Fault Tolerance (PBFT) algorithm is the first to survive Byzantine faults in asynchronous networks providing practical performance and has received considerable attention. It provides security and liveness when no more than $\lfloor \frac{n-1}{3} \rfloor$ nodes are faulty.

Fig. 1 gives a simple example of the PBFT consensus. C is a client who requests for the execution of the consensus by sending a request message to the primary, i.e., node *0*. Then node *0* starts the PBFT protocol together with the backup nodes *1-3* (here node *3* is a Byzantine node). The protocol consists of three phases: *pre-prepare*, *prepare*, and *commit*. The *pre-prepare* and *prepare* phases are used to reach an agreement on the content to be verified (named *view* in the PBFT protocol), then the *prepare* and *commit* phases are used to reach a consensus on the request and generate the reply. For more details about the protocol, [26] can be referred to.

C. Paillier Homomorphic Encryption

Homomorphic encryption is a kind of classical cryptographic algorithm, which is constructed based on intractable mathematical problems. By leveraging homomorphic encryption, the aggregated result can be obtained from the computation operation of ciphertext domain without knowing particular plaintext data. In order to implement summation operation for electronic data aggregation, we utilize a Paillier algorithm, which is an additive homomorphic encryption algorithm. In an additive homomorphic encryption, if a and b are two numbers in the plaintext domain, there are an operation $\oplus$ such that $E(a+b) = E(a) \oplus E(b)$. There are three main steps in a Paillier algorithm, i.e., key generation, encryption and decryption.

1) *Key Generation:* Choose two random large prime p and q, where $\mathsf{gcd}(pq, (p-1)(q-1)) = 1$. Then compute $n = pq$ and $\lambda = \mathsf{lcm}(p-1, q-1)$. Nextly, choose a generator $g \in Z^*_{n^2}$, and further compute $\mu = (L(g^{\lambda} \bmod n^2))^{-1}$, where $L(u) = \frac{u-1}{n}$. Now we have a public key $\mathsf{pk} = (n, g)$ and the corresponding private key $\mathsf{sk} = (\lambda, \mu)$.
2) *Encryption:* To encrypt the message $m \in Z_n$, choose a random number $r \in Z^*_n$ and then calculate the ciphertext $c = E(m) = g^m \cdot r^n \bmod n^2$.
3) *Decryption:* To decrypt the ciphertext $c \in Z^*_{n^2}$, calculate $m = D(c) = L(c^{\lambda \bmod n^2}) \cdot \mu \bmod n$.

For more information about the Paillier encryption algorithm, [27] can be referred to.

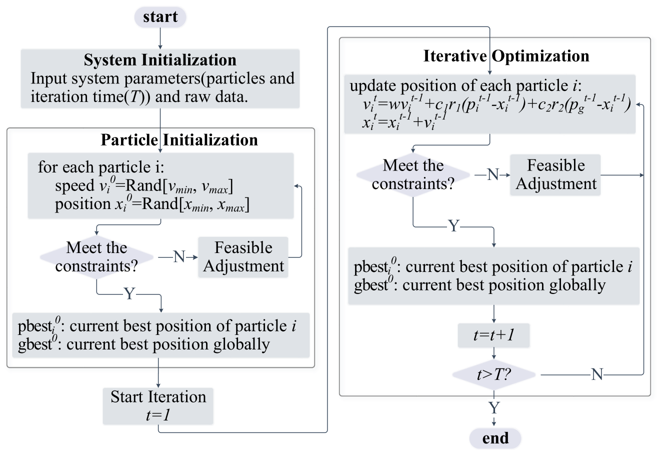
start
System Initialization
Input system parameters(particles and iteration time(T)) and raw data.
Particle Initialization
for each particle i:
speed v_i^0=Rand[v_{min}, v_{max}]
position x_i^0=Rand[x_{min}, x_{max}]
Meet the constraints?
N
Feasible Adjustment
Y
pbest$_i^0$: current best position of particle i
gbest^0: current best position globally
Start Iteration
t=1
Iterative Optimization
update position of each particle i:
$v_i^t=wv_i^{t-1}+c_1r_1(p_i^{t-1}-x_i^{t-1})+c_2r_2(p_g^{t-1}-x_i^{t-1})$
$x_i^t=x_i^{t-1}+v_i^{t-1}$
Meet the constraints?
N
Feasible Adjustment
Y
pbest$_i^0$: current best position of particle i
gbest^0: current best position globally
t=t+1
t>T?
N
Y
end

Fig. 2. Compute process of the PSO algorithm.

D. PSO Algorithm

PSO (Particle Swarm Optimization) is a kind of evolutionary computation algorithm that originates from the research in flock feeding behavior, and it aims to find the optimal solution by cooperation and information sharing among crowd individuals. In the PSO algorithm, a group of massless particles are established with two attributes to simulate birds in a flock: speed and position, where speed indicates how fast particles move and position gives their destinations. Then in the optimization process, particles are supposed to fly around in a multidimensional search space. Each particle will adjust its position according to its own experience and the shared experience of neighboring particles during flight, making use of the best position encountered by itself and its neighbors so as to find the optimal solution. The basic algorithm are described as follows.

1) *System Initialization:* As illustrated in Fig. 2, during the system initialization phase, the system parameters are initialized, e.g., the number of particles pnum, the

maximum iteration time T, and the raw data including the power usage data and weather conditions.

2) *Particle Initialization:* In the PSO initialization phase, each particle randomly generates the initial speed v_i^0 ($1 \leq i \leq \mathsf{punm}$) and position x_i^0 within the range allowed by the constraint conditions, and initializes the particle best value $\mathsf{pbest}_i^0 = v_i^0$, the global best value $\mathsf{gbest}^0 = \mathsf{pbest}_{\mathsf{imin}}^0$, where imin is the number of the particle with the least value of the pbest_i^0.

3) *Iterative Optimization:* In each iteration t, the speed of each particle v_i is updated according to the particle optimal value pbest_i and the global optimal value gbest (Eq. 1), and the current position x_i is computed according to the updated speed (Eq. 2).

$$v_i^t = wv_i^{t-1} + c_1 r_1 \left(p_i^{t-1} - x_i^{t-1}\right) + c_2 r_2 \left(p_g^{t-1} - x_i^{t-1}\right), \tag{1}$$

$$x_i^t = x_i^{t-1} + v_i^{t-1}, \tag{2}$$

where w is the inertia weight, and it can be computed as

$$w = w_{\mathsf{start}} - t(w_{\mathsf{start}} - w_{\mathsf{end}}/T).$$

The parameter t is the current iteration time and T is the maximum iteration time. The learning factors c_1 and c_2 represent the effect of local and global optimal values on particles, which can be computed as

$$c_i = \left(c_{if} - c_{ie}t/T + c_{ie}\right), i \in \{1, 2\}.$$

In general, experiences have been shown that the conditions:

$$w_{\mathsf{start}} = 0.9, w_{\mathsf{end}} = 0.4;$$
$$c_{1e} = 2.5, c_{1f} = 2.5, c_{2e} = 0.5, c_{2f} = 2.5$$

work well for most of the applications [28].

IV. System Model, Security Assumption and Design Goal

A. System Model

There are three levels of objects in our designed system: main grid, microgrids, and individual objects such as users and energy stations. These objects form a three-level control structure: internal microgrid, inter-microgrid, and main grid dispatching. For both microgrid and main grid, there are two choices for maintenance and control: centralized and distributed. Due to cost and efficiency considerations, we choose distributed control for the microgrid and centralized control for main grid, respectively.

As illuminated in Fig. 3, a microgrid system consists of a consortium and four layers.

Power Consortium: The power consortium consists of some entities with strong computing power in the microgrid, e.g., new energy power generators or buildings' administrators. Entities in the consortium is responsible for maintaining dispatching contracts as dispatchers and uploading data to the blockchain as bookkeepers(BKs).

Control Layer: The control layer is responsible for power dispatching and consists of a lot of smart contracts. The smart contracts are deployed on the Power Chain and maintained by the dispatchers. They periodically read user data from the data layer, and implement the dispatching procedure to obtain the power generation and distribution results.

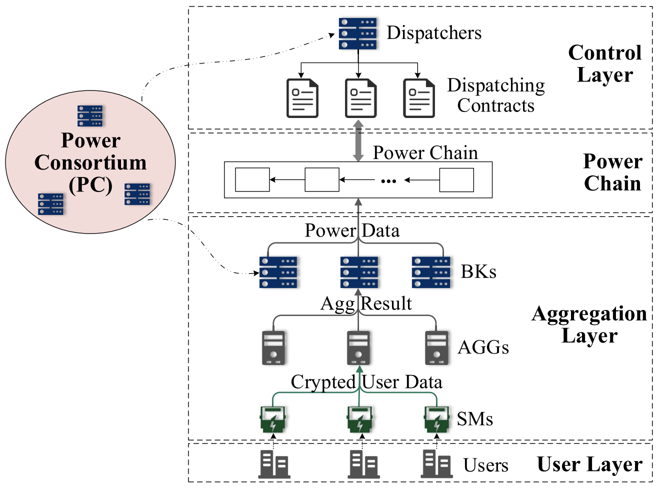

Fig. 3. The architecture of the microgrid.

Power Chain: The power chain is a blockchain responsible for the power system enabling smart contracts. As shown in Fig. 3, the power chain has two main functions: supporting the dispatching contracts and storing power data. Each microgrid has its own control layer, aggregation layer, and user layer, but all the microgrids share the same power chain. Considering the high latency of existing public blockchains such as Ethereum, the power system is better to build its own consortium blockchain to ensure the efficiency of data aggregation and power dispatching.

Aggregation Layer: The purpose of the aggregation layer is to securely upload the aggregation result of user's power consumption in a particular area to the data layer while preserving the privacy of individual users' data. The aggregation layer is mainly composed of the following entities.

- **Bookkeepers (BKs):** Each entity in the power consortium (PC) can act as a BK. The set of BKs is denoted as $\mathbb{B} = \{\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_l\}$, where l is the total number of BKs in the system. BKs connect to the upper data layer and upload the aggregated data to the blockchain through smart contracts.
- **Aggregators (AGGs):** AGGs under B_i are denoted as $\mathbb{A}_i = \{\mathcal{A}_{i1}, \mathcal{A}_{i2}, \ldots, \mathcal{A}_{im}\}$, where m is the number of AGGs. AGGs are responsible for aggregating the encrypted data upload from SMs and then sending the aggregation result to the BK.
- **Smart Meters (SMs):** The set of SMs under A_{ij} is denoted as $\mathbb{S}_{ij} = \{\mathcal{S}_{ij1}, \mathcal{S}_{ij2}, \ldots, \mathcal{S}_{ijn}\}$, where n is the number of SMs under $\mathcal{A}_{ij}$. SM is installed on the user side to obtain the user's real-time power consumption data periodically, e.g., for every 30 minutes. SM encrypts each read data by using the Paillier encryption algorithm and then sends the encryption result to the corresponding upper AGG.

User Layer: All end-users are located at the user layer. End-users' power consumption data needs to be uploaded to the data layer for user charging, power dispatching, and some other personalized services. During data uploading, every user obviously doesn't want his/her real-time power consumption data to be disclosed, thereby protecting the privacy of individuals.

B. Security Assumptions

We assume that AGGs are untrusted and can delete, append, or change the data uploaded by SMs. Considering that the AGGs run the PBFT consensus and according to the PBFT consensus algorithm's requirements, we assume that more than $\frac{2}{3}$ of participants are honest. Meanwhile, most BKs are semi-trusted, curious about user data, but honestly submit the aggregation result to the blockchain. We will show in Section VI that a malicious BK will be soon detected and punished. In addition, we assume that the AGGs and the BKs do not collude with each other. In the power dispatching phase, the dispatching program runs on the Ethereum smart contracts. As long as the adversary cannot successfully attack the Ethereum, the dispatching process can run securely.

C. Design Goal

The goal of our design aim to achieving distributed, automatic power dispatching within the microgrid. Distributed dispatching means a number of entities in the microgrid run the dispatching process together, without a control center. Since the dispatching algorithm's input is user data, we need a secure data aggregation algorithm first. And on this basis, we further design a corresponding dispatching algorithm. Here, we elaborate on the design goals from three aspects in data aggregation and power dispatching as follows.

1) User Privacy Preserving: To protect user privacy, effective privacy preserving needs to be provided during the entire energy management process. In the decentralized microgrid system where there is no trusted third party, requiring that any entity could not obtain the power consumption data and power consumption mode of a single user.

2) Malicious Aggregator Resistance and Data Integrity Protection: Since it's hard to involve a trusted data center in the microgrid system, the existing schemes of identifying malicious aggregators through the data center is no longer feasible. A reliable solution is needed to resist malicious aggregators, prevent malicious or compromised aggregators from submitting forged user data, ensure the correctness of data aggregation results, and ensure the reliability of subsequent power dispatching.

3) Blockchain-Based Automatic and Distributed Power Dispatching: The most crucial goal of power dispatching is to figure out the optimal power distribution plan. Additionally, this procedure is supposed to run in a distributed, automatic, secure, and reliable way. Distributed way means many entities in the system execute the dispatching process simultaneously, without a control center. An automatic way means that this procedure can be triggered and run automatically at the appropriate time without human control or intervention.

V. Proposed Solution

A. Overview

The whole microgrid power management process can be divided into two key steps: data aggregation and power dispatching. Entities (e.g., BKs, AGGs and SMs) in the aggregation layer aggregate the user data and upload it to the data layer. Then dispatchers in the dispatching layer take the data as input and run the dispatching procedure to optimize the system operation.

To realize secure data aggregation, we exploit consortium blockchain with the PBFT consensus mechanism to establish a power data blockchain for a electric system. The aggregation process is divided into three phases: data reading and encryption, data aggregation and consensus, and data decryption and uploading. At first, SM reads the power usage data and the relevant information of a single user, encrypts it using a homomorphic encryption algorithm, signs the message, and then sends the ciphertext and signature to its associated AGG. AGG then aggregates the ciphertext periodically, executes the PBFT consensus algorithm, and sends the aggregation result to its responsible BK when reaching a PBFT consensus. Finally, BK decrypts received aggregated data and uploads the decrypted power data to the data layer. Afterward, dispatching smart contracts read the aggregation data automatically and predict future dispatching data. For power dispatching, we leverage the PSO algorithm and smart contracts to realize automatic distributed dispatching. As introduced in Section III, PSO can be utilized as an optimization algorithm for obtaining the optimal power dispatching results. Meanwhile, in order to implement the procedure automatically, a group of smart contracts, including some Particle-Contracts and an Update-Contract will execute the dispatching algorithm together.

B. System Initialization

1) Microgrid and Power Consortium Construction: The microgrids' construction depends on the geographical scope and the power supply and demand relationship. A microgrid usually consists of some neighboring residential buildings, office buildings, and new energy power stations. The division of microgrids can be assigned by the upper-level manager or negotiated between entities. When the power supply and demand relationship changes, for example, when the supply in microgrid exceeds demand while a neighboring microgrid is on the contrary, the two microgrids can exchange some users or power stations so that they can both reach a balance of power supply and demand.

When the scope of a microgrid is determined, the first thing to do is to determine the Power Consortium. The members of the consortium are usually the managers of entities, such as buildings or power stations' administrators. They conduct off-chain negotiations to determine the consortium members. After that, one of the members creates a *Microgrid Contract* on the Power Chain, recording consortium members, microgrid scope and other related information. Besides, entities' identity information (i.e., identifier, public key) will be uploaded into the *Microgrid Contract* for identity authentication. Then, all consortium members lock the deposit in the *Microgrid*

Contract. In this way, the microgrid's construction is completed. Besides, it does not matter which member is to release the *Microgrid Contract*. Because if the released contract is incorrect, other members will not lock the deposit in the contract; thus, the contract represents a microgrid consists of the member itself, which is meaningless.

2) SM/AGG Assignment and Key Registration: When a microgrid is successfully constructed, the data aggregation structure is to be divided based on the geographical location. For example, smart meters in a building are assigned to the aggregator served by the building administrator. And suppose that there are l bookkeepers in the microgrid, then the aggregators are divided into l groups and each group is assigned to a bookkeeper. To prevent collusion, the assignment between aggregators and bookkeepers can be updated regularly.

After the assignment, entities upload their public keys and identifiers to the *Microgrid Contract*. Then in the data aggregation process, they can get the required information from the Power Chain. Each bookkeeper is supposed to generate a Paillier homomorphic key pair ($\mathsf{sk}_i = (\lambda, \mu), \mathsf{pk}_i = (n, g)$) (as explained in Section III-C) and upload the public key to the *Microgrid Contract*. Each aggregator is supposed to generate an AGG quadruple ($\mathsf{Prk}_{ij}, \mathsf{Puk}_{ij}, \mathsf{Id}_{ij}, r$), where ($\mathsf{Prk}_{ij}, \mathsf{Puk}_{ij}$) is a key pair for digital signature and authentication, Id_{ij} is a random unique identifier, and r is a random number used for homomorphic encryption. Then the aggregator uploads ($\mathsf{Puk}_{ij}, \mathsf{Id}_{ij}, r$) to the *Microgrid Contract*. Each user is also supposed to generate a SM triple ($\mathsf{Prk}_{ijk}, \mathsf{Puk}_{ijk}, \mathsf{Id}_{ijk}$), where the key pair is similar to the aggregators', and Id_{ijk} is a unique identifier consists of Id_{ij} (to indicate the assigned AGG) and a random identifier (for unique number). Then the user imports the triple into the smart meter and uploads ($\mathsf{Puk}_{ijk}, \mathsf{Id}_{ijk}$) to the *Microgrid Contract*.

C. Secure Data Aggregation

The data aggregation process consists of three phases. The relevant details are described as follows.

1) Data reading and encryption: Each smart meter $\mathcal{S}_{ijk}(1 \le k \le n)$ reads the power consumption data pd_{ijk} periodically and encrypts it with BK $\mathcal{B}_i$'s Paillier homomorphic public key $\mathsf{pk}_{\mathcal{B}_i} = (n, g)$ and a random number r generated by $\mathcal{A}_{ij}$ in the system initialization stage. For data aggregation, smart meters assigned to one AGG share the same r. We use cpd_{ijk} to denote the encryption result, then we have $\mathsf{cpd}_{ijk} = \mathsf{E}(\mathsf{pk}_{\mathcal{B}_i}, \mathsf{pd}_{ijk}) = g^{pd_{ijk}} \cdot r^n$. Next, $\mathcal{S}_{ijk}$ generates the *Raw-Data-Message* as

$$\mathsf{RDmsg}_{ijk} = \{\mathsf{cpd}_{ijk}||\mathsf{Id}_{ijk}||\mathsf{date}_{ijk}||\mathsf{ts}_{ijk}\},$$

where Id_{ijk} is the identifier number of $\mathcal{S}_{ijk}$, date_{ijk} is the date information, and ts_{ijk} is the timestamp. Then, $\mathcal{S}_{ijk}$ signs the *Raw-Data-Message* RDmsg_{ijk} and get the signature RDsig_{ijk}. Finally, $\mathcal{S}_{ijk}$ sends $\{\mathsf{RDmsg}_{ijk}||\mathsf{RDsig}_{ijk}\}$ to its corresponding AGG $\mathcal{A}_{ij}$.

2) Data aggregation and consensus: AGG $\mathcal{A}_{ij}$ is supposed to collect messages from SM which it is responsible for, i.e., $\mathcal{S}_{ij1}, \mathcal{S}_{ij2}, \ldots, \mathcal{S}_{ijn}$, and verify the corresponding signature $\mathsf{RDsig}_{ijk}(1 \le k \le n)$ to check whether the message is uploaded by a legal SM. Besides, consider that some SMs may fail and cannot upload data on schedule or upload invalid data (e.g., with invalid signatures). Under the circumstances, the AGG ignores these faulty SMs and performs aggregation process only on the valid SMs' data. For clarity, we use $\mathbb{VS}_{ij}$ to denote the set of the valid SMs' under $\mathcal{A}_{ij}$. According to the homomorphic encryption' features, $\mathcal{A}_{ij}$ then aggregates users' power data in the *Raw-Data-Messages* and gets the *encrypted-Aggregation-Result* cAggRS_{ij} as

$$\mathsf{cAggRS}_{ij} = \oplus_k \mathsf{cpd}_{ijk}, \mathcal{S}_{ijk} \in \mathbb{VS}_{ij}.$$

Then $\mathcal{A}_{ij}$ further generates the *Message-to-be-Confirmed* TBCmsg_{ij} as

$$\mathsf{TBCmsg}_{ij} = \mathbb{VS}_{ij}||\mathsf{cAggRS}_{ij}||\mathsf{Id}_{ij}||\mathsf{date}_{ij}||\mathsf{ts}_{ij},$$

where Id, date and ts represent the identifier number, date and timestamp, respectively. After that, $\mathcal{A}_{ij}$ signs TBCmsg_{ij} and get the signature TBCsig_{ij}. Then, $\mathcal{A}_{ij}$ initiates the PBFT consensus to prove the correctness of the aggregation result cAggRS_{ij} by broadcasting the relevant information (explained in the *Pre-Prepare* stage) within the consensus group $\mathbb{A}_i$ (m AGGs under the same BK $\mathcal{B}_i$). Here we should note that each AGG initiates a PBFT consensus once as a master node and participate m times as a common node.

As shown in Fig. 4, the PBFT consensus process consists of three main stages: *Pre-Prepare*, *Prepare* and *Commit*:

- **Pre-Prepare:** The *Pre-Prepare* stage is used to initiate the PBFT consensus process. On the *Pre-Prepare* stage, the master node $\mathcal{A}_{ij}$ generate the *Pre-Prepare-Message* with three contents: 1) $\{\mathbb{VS}_{ij}||(\mathsf{RDmsg}_{ijk}||\mathsf{RDsig}_{ijk})(\mathcal{S}_{ijk} \in \mathbb{VS}_{ij})\}$, which is the message sent from SMs, 2) the *Aggregation-Result* and relevant information TBCmsg_{ij}, and 3) the signature TBCsig_{ij}. Then $\mathcal{A}_{ij}$ broadcast the *Pre-Prepare-Message* to all other AGGs (called common nodes in PBFT algorithm) in the charge of $\mathcal{B}_i$, i.e., $\mathcal{A}_{ij'}$, $1 \le j' \le m, j' \ne j$.
- **Prepare:** The *Prepare* stage is used for aggregators to verify and declare the correctness of the aggregation result. After receiving the *Pre-Prepare-Message* from the

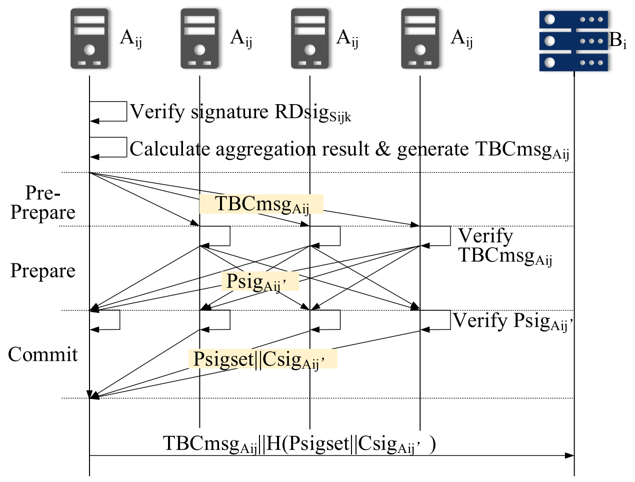

Fig. 4. PBFT consensus for data aggregation.

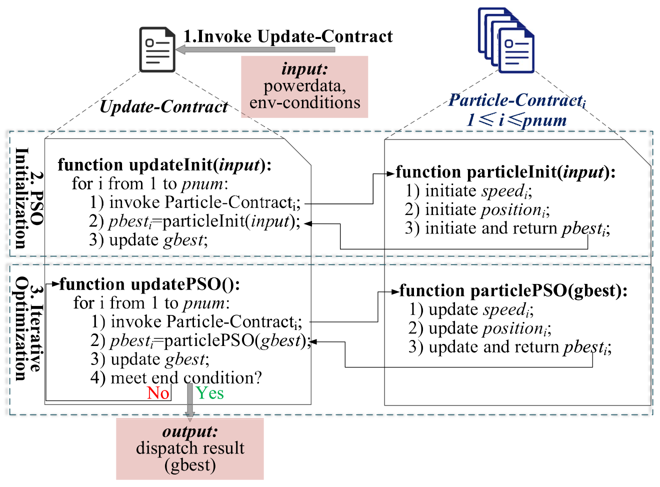
1.Invoke Update-Contract
input: powerdata, env-conditions
Update-Contract
Particle-Contract$_i$ $1\leq i\leq pnum$
2. PSO Initialization
function updateInit(input):
for i from 1 to pnum:
1) invoke Particle-Contract$_i$;
2) pbest$_i$=particleInit(input);
3) update gbest;
function particleInit(input):
1) initiate speed$_i$;
2) initiate position$_i$;
3) initiate and return pbest$_i$;
3. Iterative Optimization
function updatePSO():
for i from 1 to pnum:
1) invoke Particle-Contract$_i$;
2) pbest$_i$=particlePSO(gbest);
3) update gbest;
4) meet end condition?
No
Yes
function particlePSO(gbest):
1) update speed$_i$;
2) update position$_i$;
3) update and return pbest$_i$;
output: dispatch result (gbest)

Fig. 5. The dispatching contracts consist of the update-contract and some particle-contracts.

master node, $\mathcal{A}_{ij'}$ first verify $\mathsf{RDsig}_{ijk}(\mathcal{S}_{ijk} \in \mathbb{VS}_{ij})$ to confirm the correctness of the data source. If correct, $\mathcal{A}_{ij'}$ then calculates $\mathsf{cAggRS}' = \oplus_k \mathsf{cpd}_{ijk}(\mathcal{S}_{ijk} \in \mathbb{VS}_{ij})$ and determines whether cAggRS' is equal to cAggRS. If equal, $\mathcal{A}_{ij'}$ now enters *Prepared-State*, then signs TBCmsg_{ij} to get the *Prepared-Signature* $\mathsf{Psig}_{ij'}$ and broadcasts it. In brief, entering the *Prepared-State* means that $\mathcal{A}_{ij'}$ declares that he considers the aggregation result from $\mathcal{A}_{ij}$ is correct and waits for the verification results of other aggregators.

- **Commit:** The *Commit* stage is used to reach the PBFT consensus on the total system status. On the *Commit* stage, the master node $\mathcal{A}_{ij}$ and also common nodes $\mathcal{A}_{ij'}$ collect the *Prepared-Signatures* $\mathsf{Psig}_{ij''}$, $1 \leq j'' \leq m$. For the description, assume that $m = 3f+1$. Then, once $\mathcal{A}_{ij}$ or $\mathcal{A}_{ij'}$ collect more than $2f+1$ *Prepared-Signature*s, they then enter the *Commited-State*. Afterward, for a common node $\mathcal{A}_{ij'}$, it generates a signature set $\mathsf{Psigset}$ from the collected signatures. Then $\mathcal{A}_{ij'}$ signs $\mathsf{Psigset}$ and gets the *Commited-Signature* $\mathsf{Csig}_{ij'}$, and the *Commit-Message* is $\mathsf{Psigset}||\mathsf{Csig}_{ij'}$. Finally, $\mathcal{A}_{ij'}$ send the *Commit-Message* to the master node $\mathcal{A}_{ij}$. Unlike the common nodes, however, the master node $\mathcal{A}_{ij}$ do not need to generate the *Commit-Message*. When entered into the *Committed-State*, $\mathcal{A}_{ij}$ only waits for the *Commit-Message*s sent from other nodes. Once $\mathcal{A}_{ij}$ collects $2f$ *Commit-Message*s, the system reaches a PBFT consensus.

The significance of the *Pre-Prepare* and *Prepare* stage is relatively obvious, we need to specially explain why the *Commit* stage is necessary. If we remove the commit stage, $\mathcal{A}_{ij'}$ verify the aggregation result and send the declaration to $\mathcal{A}_{ij}$. In this case, only $\mathcal{A}_{ij}$ knows the consensus result, each AGG only has partial system information. However, with commit stage, $\mathcal{A}_{ij'}$ not only verifies by itself, but also refers to other AGGs' verification results. In this way, all honest AGGs' assertions on all aggregated results are consistent. Besides, the honest AGGs' have consistent perceptions of which AGGs are Byzantine at this moment. This provides the possibility for some system requirements, such as finding and correcting the wrong AGGs.

When reaching the PBFT consensus, $\mathcal{A}_{ij}$ then uploads TBCmsg_{ij} and the *Commit-Message*s $\mathsf{Psigset}||\mathsf{Csig}_{ij'}$ to the BK $\mathcal{B}_i$.

3) Data decryption and uploading: After receiving messages uploaded from $\mathcal{A}_{ij}$, $1 \leq j \leq m$, BK $\mathcal{B}_i$ first validates the *Commit-Messages*s to determine whether the aggregation result is trusted. If there are no problems, $\mathcal{B}_i$ then decrypts the *encrypted-Aggregation-Result* to acquire the aggregation result $\mathsf{AggRS}_{ij} = \Sigma_{k=1}^{n} \mathsf{pd}_{ijk}$, and then upload the decrypted data and the digest of relevant verification information (i.e., *Commit-Messages*) to the blockchain. Except for the aggregation result, the data uploaded to the blockchain also contains the valid SM set $\mathbb{VS}$, which can help the power system's administrators to check which SMs are faulty (not in $\mathbb{VS}$) and notify the corresponding users to repair the SMs. Besides, this also prevents AGGs from maliciously discarding some SMs' data.

D. Dispatching Over Smart Contract

To use the PSO algorithm for power dispatching, we model the power system to evaluate the cost of each power generator. The objective function of the dispatching algorithm is the total system cost. Afterward, under the constraints of power balance, capacity limitation, and other restrictions, the objective function's optimal value representing the minimum system cost is computed.

On the foundation of the secure public data aggregation system proposed in Section V-C, we realize the distributed automatic dispatching in a microgrid through smart contracts. Participants, including power stations, positive users, and BKs within the microgrid, will contribute to the deployment and maintaining of the dispatch program. At the initially establishing stage of the dispatch contract, the PSO algorithm and relevant parameters will be written into the smart contracts as initial parameters. Afterward, the distributed power dispatching system now exists in the blockchain in the form of smart contracts.

1) The Dispatching Contracts: Considering the computing power limitation of the smart contract, we distribute the dispatching process to multiple smart contracts, and they coordinate to complete the dispatching process. The dispatching process consists of two kinds of smart contract: some *Particle-Contract*s and an *Update-Contract*. Each *Particle-Contract$_i$* represents a particle i in the PSO algorithm, and the *Update-Contract* is responsible for global adjustment. Similar to the PSO algorithm, the dispatching process is divided into three main step.

- **Invoke update-contract:** To start a dispatching process, entities in blockchain need to invoke the *update-contract*, and take the user data, environmental conditions and other relevant information as input.
- **PSO initialization:** The *Update-Contract* will first call the *updateInit*(input) function to initiate the PSO algorithm. The input is the relevant information inputted in the invoking stage. The *updateInit* function invokes each *Particle-Contract$_i$* ($1 \leq i \leq \mathsf{pnum}$) in turn, and call the *particleInit*(input) function to initiate each particle, including the speed_i, $\mathsf{position}_i$ and pbest (particle best

value). Then the *particleInit* function will return pbest_i, and the *updateInit* function update gbest (global best value) according to it.

- **Iterative optimization:** After PSO initialization, the *Update-Contract* will execute the *updatePSO*() function, iteratively invokes each *Particle-Contract$_i$* and passes gbest to the *particlePSO* function. The *particlePSO* function firstly updates speed_i according to gbest, then updates the $\mathsf{position}_i$ on the basis of speed_i, and finally updates and returns the particle best value pbest. The *Update-Contract$_i$* then update the global best value gbest according to all the pbest_i $(1 \leq i \leq \mathsf{pnum})$. After each iteration is completed, gbest represents the system's current optimal solution.

When reaching the maximum iteration time or the optimal solution threshold, the scheduling algorithm then terminates, and the *Update-Contract* then returns the dispatching result.

2) Deployment of the Dispatching Contracts: In a microgrid, all power generation and distribution process within a period of time should follow the same rules. Therefore, there is only one correct dispatching contract in each period. The dispatching rules (including a security factor ϵ, $0 < \epsilon \leq 1$) are negotiated and decided by the power consortium's members, and this process is unrelated to the blockchain. After reaching an agreement on the dispatching rules, one of the members converts the rules into smart contract codes and publishes them on the blockchain. The dispatching contract will trigger an activation challenge when first released. Then other members respond to the challenge to activate the dispatch contract (see Section V-D for more details). The contract is activated after receiving more than ϵl (l is the number of dispatchers in the power consortium) responses and starts the dispatching process. If the contract does not receive enough responses within the time limit, it will close, and dispatchers need to reissue a new dispatching contract.

VI. Security Analysis

To illustrate the security features of our proposed microgrid system, we analyze the aggregating and dispatching process.

A. User Privacy Preserving

Theorem 1: Eavesdropping on the user side cannot reveal individual user data.

Proof: In the data reading and encryption phase in data aggregation, the user $\mathcal{S}_{ijk}$'s power data $\mathsf{pd}_{\mathcal{S}_{ijk}}$ is encrypted to cpd_{ijk} by $\mathsf{cpd}_{ijk} = \mathsf{E}(\mathsf{Puk}_{\mathcal{B}_i}, \mathsf{pd}_{\mathcal{S}_{ijk}})$. Thus, only the BK $\mathcal{B}_i$ can reveal pd_{ijk} from cpd_{ijk}. We assume that an adversary $\mathbf{A}$ can lurk on the user side and eavesdrop the data communicated between SM and AGG. As long as $\mathbf{A}$ is not $\mathcal{B}_i$, it cannot decrypt cpd_{ijk}. However, we propose in the security assumption that $\mathcal{B}$ is curious but honest, so it will not actively eavesdrop data on the user side. Thus, no attacker can obtain the actual data of a single user. Therefore, user privacy can be effectively preserved. ■

Theorem 2: Compromising a BK cannot reveal individual user data.

Proof: After receiving messages sent from AGGs $\mathcal{A}_{ij}(1 \leq j \leq m)$, BK $\mathcal{B}_i$ can acquire the aggregation result $\mathsf{AggRS}_{\mathcal{A}_{ij}} = \Sigma_{k=1}^{n} \mathsf{pd}_{\mathcal{S}_{ijk}}$ by decrypting the *Aggregation-Result* $\mathsf{cAggRS}_{\mathcal{A}_{ij}}$. Although the adversary $\mathbf{A}$ successfully compromised the BK $\mathcal{B}_i$ and accessed its database, $\mathbf{A}$ could only get the aggregation result AggRS_{ij} which is the sum of n user's data, but cannot reveal the specific data of an individual user $\mathcal{S}_{jik}$. ■

B. Malicious Aggregator Resistance and Data Integrity Protection

Theorem 3: Compromising less than $\frac{1}{3}$ AGGs cannot tamper with user data.

Proof: Suppose that an adversary $\mathbf{A}$ can either masquerade as a smart meter $\mathcal{S}_{ijk}$ to upload false user data cpd'_{ijk}, or compromise an aggregator $\mathcal{A}_{ij}$ to tamper with cpd_{ijk} to cpd'_{ijk}, and then lead to a false aggregation result $\mathsf{cAggRS}'_{\mathcal{A}_{ij}} = \mathsf{cpd}_{ij1} \oplus \cdots \oplus \mathsf{cpd}'_{ijk} \oplus \cdots \oplus \mathsf{cpd}_{ijn}$. In the consensus process's *prepare* phase, other AGGs $\mathcal{A}_{ij'}(1 \leq j' \leq m, j' \neq j)$ will verify each $\mathsf{RDsig}_{\mathcal{S}_{ijk}}$ to determine cpd_{ijk}'s validity, but $\mathbf{A}$ cannot provide a valid $\mathsf{RDsig}'_{\mathcal{S}_{ijk}}$ for cpd'_{ijk} due to the lack of $\mathcal{S}_{ijk}$'s private key $\mathsf{Prk}_{\mathcal{S}_{ijk}}$. Thus, the false data will not reach the PBFT consensus and thus cannot be uploaded to the blockchain. Hence compromising no more than f ($m = 3f + 1$) AGGs cannot tamper with user data. ■

Theorem 4: Compromising an AGG to deliberately discard some SMs' power data can be detected.

Proof: Suppose that an adversary $\mathbf{A}$ can compromise an AGG $\mathcal{A}_{ij}$ to deliberately discard a SM's power data pb_{ijk}. In this case, the aggregation process will be executed normally, and pb_{ijk} is not contained in the final aggregation result. However, as thus, k will also be not in the valid SM set $\mathbb{VS}$, and the administrators will consider $\mathcal{S}_{ijk}$ as faulty and notify the corresponding user to repair the SM. If they find that $\mathcal{S}_{ijk}$ has been operating normally, they can determine that $\mathcal{A}_{ij}$ is compromised and perform the corresponding process. ■

Theorem 5: Masquerading as a legal SM cannot submit false user data.

Proof: An adversary $\mathbf{A}$ may try to masquerade as a SM $\mathcal{S}_{ijk'}$ to upload false user data $\mathsf{pd}_{\mathcal{S}_{ijk'}}$ to disturb the system's regular operation. However, in the data reading and encryption phase, each SM $\mathcal{S}_{ijk}$ will sign the encrypted data before uploading. And in the data aggregation and consensus phase, the AGG $\mathcal{A}_{ij}$ will first verify the signature to determine whether the data is from a legal SM. Thus, data submitted by an illegal SM will be dropped directly, and this kind of attack will not succeed. ■

Theorem 6: Compromising a BK to upload false data can be detected.

Proof: Suppose that an adversary $\mathbf{A}$ can compromise a bookkeeper $\mathcal{B}_i$ to upload false aggregation data AggRS'_{ij}. Although this type of attack cannot be prevented in advance, any entity can soon detect it. In the data decryption and uploading phase, the digest of the relevant verification information (including AGGs' PBFT signatures $\mathsf{PSigset}$ and $\mathsf{CSig}_{\mathcal{A}_{ij}}$) need to be uploaded together with AggRS_{ij}. However, $\mathbf{A}$ can only generate AggRS'_{ij} but cannot calculate the corresponding $\mathsf{PSigset}$ and $\mathsf{CSig}_{\mathcal{A}_{ij}}$ due to the lack of the AGGs' private keys for signature. Therefore, once $\mathbf{A}$ uploads the data that has been tampered with, any entity in the system can detect that the signatures ($\mathsf{PSigset}$ and $\mathsf{CSig}_{\mathcal{A}_{ij}}$) are not consistent with the data, thereby determining that BK $\mathcal{B}_i$ has been attacked by an adversary $\mathbf{A}$, and perform the corresponding process. ∎

C. Security of Automatic and Distributed Power Dispatching

Theorem 7: Compromising less than $(1-\epsilon)\cdot l$ dispatchers cannot corrupt the dispatching procedure.

Proof: The dispatching process runs on the blockchain. Due to the security of the blockchain, we can assert that as long as a correct set of dispatching contracts is deployed, it can run in accordance with the expected steps. However, a compromised dispatcher may deploy malicious dispatching contracts with wrong optimal algorithms, which may lead to false dispatching results. The security of the dispatching algorithm is determined by the security factor ϵ; that is, the algorithm can tolerate less than $(1-\epsilon)\cdot l$ malicious dispatchers. The value of ϵ is related to the trust among consortium members and is jointly determined by the members. For example, when the proportions of power station-side dispatchers and user-side dispatchers in the consortium are similar, it is reasonable to set $\epsilon = \frac{1}{2}$. ∎

D. Comparison With Existing Data Aggregation Solutions

As mentioned in Section II, there are several solutions for smart grid data aggregation, including Li's solution [6], EPPA [7], PPMA [12], and Ghadamyari's solution [14]. As shown in Table I, we compare the solutions from five aspects: privacy preserving, malicious AGG resistance, malicious BK/DC resistance, no trusted third party, and trusted sharing. Privacy preserving is the most basic requirements of a data aggregation scheme, and is provided by data encryption. Malicious AGG resistance protects data integrity, which requires that if there are malicious aggregators (or aggregation gateways) that attempt to tamper with, delete, or forge data, this malicious behavior can be discovered. Similarly, malicious BK/DC resistance requires that if some bookkeepers (or the data center) is compromised and try to tamper with, delete, or forge data, this malicious behavior can be discovered. No trusted third party means that entities generate their security parameters themselves, without the need for a trusted third party. Trusted sharing means that entities without mutual trust can share their data reliably, that is, the shared data is correct.

VII. Performance Analysis

In this section, we first implement the proposed data aggregation scheme by python3 to conduct its performance analysis. To get close to the actual computing power of the aggregators in microgrids, we produce our experiments on the Raspberry Pi with ARM1776 CPU and 256 MB memory. Furthermore, we implement the power dispatching algorithm by Solidity and test it on Ethereum test network. We verify the dispatching effectiveness and estimate the computation overhead measured in gas consumption.

TABLE I
Security Functions of Data Aggregation Solutions

DataAgg scheme	Privacy Preserving	Malicious AGG Resistance	Malicious BK/DC Resistance	No Trusted Third Party	Trusted Sharing
Li's solution	✓				
EPPA	✓	✓			
PPMA	✓	✓			
Ghadamyari's solution	✓		✓	✓	✓
Our scheme	✓	✓	✓	✓	✓

TABLE II
Computation Overhead of the Data Aggregation Process

Data Read	RDmsg	RDsig	Data Agg	ver RDsig	TBCmsg	TBCsig
	C_p	C_{sig}		nC_{sig}	C^n_{Agg}	C_{sig}
Consensus	Pre-Prepare	Prepare			Commit	
	broadcast	ver RDsig	ver cAggRS	Psig	ver Psigset	Csig
	/	$m(n-1)C_{\text{sig}}$	mC^n_{Agg}	mC_{sig}	$(m-1)C_{\text{sig}}$	C_{sig}

A. Computational and Communication Overhead of Data Aggregation Scheme

Suppose that the computational cost for Paillier encryption or decryption is C_P, for signature or verification is C_{Sig}, and for aggregation of n ciphertexts is C^n_{Agg}. Table II shows the computational overhead of our proposed data aggregation scheme. The data aggregation scheme consists of three phases, which are respectively completed by SM, AGG, and BK. In the data reading and encryption phase, a SM need to execute one encryption and one signature operation. Thus, the total cost of a SM is C_P+C_{Sig}. In the data aggregation and consensus phase, AGG $\mathcal{A}_{ij}(1\leq j\leq m)$ first verifies RDsig, then aggregate the received SMs' power data and generate TBCmsg, and finally sign it. Therefore, the computational overhead is $nC_{\mathsf{sig}}+C^n_{\mathsf{Agg}}+C_{\mathsf{sig}}$. Then, each AGG executes once as the PBFT master node and $m-1$ times as a PBFT common node. As the master node, $\mathcal{A}_{ij}$ first verifies $\mathcal{S}_{ijk}(1\leq k\leq n)$'s signatures, then aggregate $\mathsf{cpd}_{ijk}(1\leq k\leq n)$, and afterward initiate the PBFT consensus process. In the *Pre-Prepare* stage, $\mathcal{A}_{ij}$ broadcasts the *Pre-Prepare-Message*. In the *Prepare* stage, each common node verify $\mathcal{S}_{ijk}(1\leq k\leq n)$'s signatures, aggregate $\mathsf{cpd}_{ijk}(1\leq k\leq n)$, and sign the aggregation result. Therefore, the overhead is $m(nC_{\mathsf{sig}}+C^n_{\mathsf{Agg}})$ In the *Commit* stage, each node need to verify and sign the $\mathsf{Psigset}$, and the overhead is mC_{sig}. Finally, in the data decryption and uploading phase, BK need to validate $Psig_{ij}$ and $\mathsf{Csig}_{ij}(1\leq j\leq m)$ firstly, then decrypt $\mathsf{cAggRS}_{ij}(1\leq j\leq m)$, and finally sign the decryption result and upload it to the blockchain.

We implement the data aggregation process and test the runtime under different number of users on the Raspberry Pi with ARM1776 CPU and 256 MB memory. The total number of users is $l\cdot m\cdot n$, where l is the number of bookkeepers in the microgrid, m represents the number of aggregators under the same bookkeeper and running the PBFT process together, and n represents the number of smart meters that each aggregator is responsible for. We test the runtime to aggregate $m\cdot n$ users' data under one bookkeeper. As Fig. 6 shows, the runtime has a

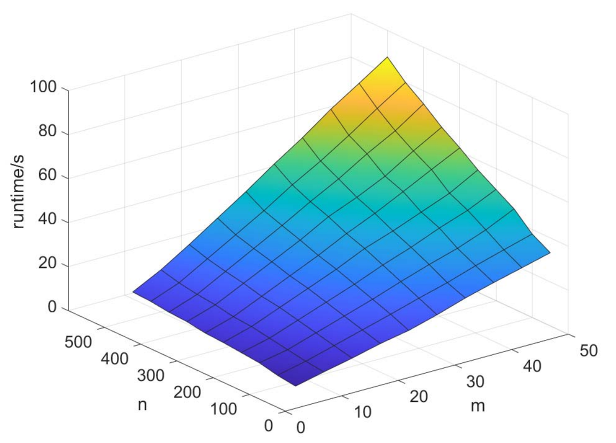

Fig. 6. The runtime of the proposed data aggregation scheme with different m and n.

linear relation with both m and n, which is consistent with the above analysis. When $m = 50$ and $n = 500$, a bookkeeper can aggregate $50 \cdot 500 = 25,000$ users' data within 90 seconds. Since all the aggregation processes under the l bookkeepers are parallel, the whole microgrid can aggregate $25,000 \cdot l$ users' data within 90s, perfectly achieving the practically applied request.

Besides, suppose that the average communication overhead between two aggregators is C_{com}. The arrows in Fig. 4 represent the communications between AGGs. In the *Pre-Prepare* stage, the master node $\mathcal{A}_{ij}$ broadcast the *Pre-Prepare-Message* to other AGGs, and the communication overhead is $(m-1)C_{\mathsf{com}}$. In the *Prepare* stage, each AGG except for the master node need to make a broadcast, and cause $(m-1)^2C_{\mathsf{com}}$ communication overhead. In the *Commit* stage, each AGG send a message to the master node, and the overhead is $(m-1)C_{\mathsf{com}}$. Thus, the total communication overhead of the consensus process is $(m^2-1)C_{\mathsf{com}}$.

B. Dispatching Effects of Power Dispatching Scheme

To effectively evaluate the dispatching contracts' running cost, we complete the dispatching contracts using Solidity and deploy them on an Ethereum test blockchain. We set the user power consumption and natural environments (e.g., light intensity and wind power) within 24 hours of a day and run dispatching contracts each hour. The power consumption of users is between 40-150 kW, and the power of the generators is 50 kW. The cost of power generation is calculated according to practice standards, including the construction, maintenance, pollution treatment, etc., and the unit is 1, 000\$. In fact, we don't need to pay much attention to the absolute power generation cost, cause there are various cost calculation methods. What is important is the optimization that power dispatching can bring. For power dispatching, we invoke the *Update-Contract* and input the sum of user power data, light intensity, and wind speed (as environmental conditions) for each hour. The *Update-Contract* then run the dispatching program to compute the optimal generation amount of each power

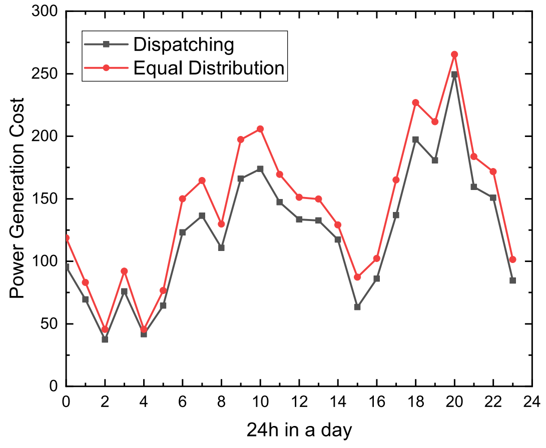

Fig. 7. The power dispatching's impact on the power generation cost in a day.

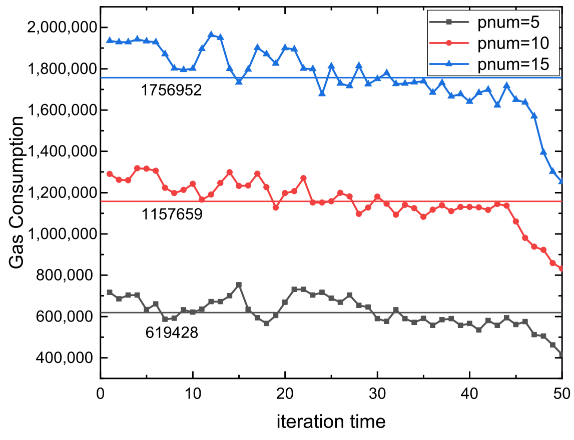

Fig. 8. Gas consumption of the dispatching contract with different particle numbers and iteration times.

generator to make the total cost lowest. We record the dispatching result when $\mathsf{pnum} = 10$ and $t = 50$. Moreover, we use an equal distribution that evenly distributes power demand to each power station to validate the dispatching effects. Fig. 7 shows the dispatching result for 24 hours of a day, the x-axis represents the 24 hours of the day, and the y-axis represents the dispatching overhead. The circular icon line represents the cost of equal distribution (means evenly distributes power demand to each power station), and the line of the square icon is the cost of running the dispatching algorithm. It is clear that running power dispatching can significantly reduce the system cost.

C. Running Cost of Power Dispatching Scheme

The running cost of Ethereum smart contracts can be measured by gas consumption. We test the gas consumption with different particle number pnum (5, 10, 15) and iteration time t ($1 \le t \le 50$). By splitting the PSO algorithm into multiple smart contracts and execute together, the gas consumption is controlled within the gas limited range, and the dispatching algorithm can be executed through the smart contract. As Fig. 8 shows, when pnum is 5, 10, and 15, the average gas consumption is 619, 428, 1, 157, 659, and 1, 756, 952. The gas consumption increases as pnum increases and is roughly linearly related. As for a fixed pnum, as t increases, the gas consumption fluctuates around a higher value at an earlier stage, then slowly decreases, and finally drops significantly somewhere. This is because the result of each iteration will be closer to the optimal solution. Thus the computation cost of the next iteration will be lower. When a particle reaches the optimal solution, its cost reaches the minimum and no longer drops. As the iteration time increases, more particles reach the optimal solution and then the total overhead begins to decrease significantly, and eventually, all particles reach the optimal solution, and the total gas cost will stabilize at a lower value. However, the iterative process does not wait until all particles reach the optimal solution, but as long as one particle reaches the optimal value, we can then get the system optimal value. Therefore, when the gas consumption drops significantly, the optimal solution has been found (actually earlier, but the impact of a single particle reaching the optimal value on the total cost is not obvious). In addition, the more particles there are, the more obvious the downward trend of gas will be. This is because the algorithm can find the optimal solution faster when the particle number is large; and since each particle's computation cost will reduce, the total cost will naturally reduce more obviously as the particle number increases.

It should be noted that we chose Ethereum smart contract for experiments because Ethereum is widely accepted and the contracts' computation overhead can be measured by gas consumption, so as to better evaluate the performance of our proposed dispatching scheme. In practice, the power system can build its own *Power Chain* (as shown in Fig. 3) instead of directly using the expensive Ethereum. Therefore, the actual price of Ether that the experiment results show is not important.

VIII. Conclusion

In this paper, by leveraging the blockchain technology, we proposed an decentralized energy management solutions for microgrids, which have two innovation parts. 1) A data aggregation scheme based on the Paillier homomorphic encryption and the PBFT consensus algorithm, which can guarantee the correctness of aggregation results in a decentralized environment without any authority third party. 2) An automatic and distributed power dispatching scheme by utilizing Ethereum smart contracts and the PSO algorithm. Through the security analysis and the experimental implementation, we prove that the proposed solution can well achieve all the required security features while providing high efficiency and guaranteed correctness of aggregation results.

References

[1] F. F. Wu, K. Moslehi, and A. Bose, "Power system control centers: Past, present, and future," *Proc. IEEE*, vol. 93, no. 11, pp. 1890–1908, Nov. 2005.

[2] F. R. Badal, P. Das, S. K. Sarker, and S. K. Das, "A survey on control issues in renewable energy integration and microgrid," *Prot. Control Mod. Power Syst.*, vol. 4, no. 1, pp. 1–27, 2019.

[3] S. Anand and B. G. Fernandes, "Reduced-order model and stability analysis of low-voltage DC microgrid," *IEEE Trans. Ind. Electron.*, vol. 60, no. 11, pp. 5040–5049, Nov. 2013.

[4] R. H. Lasseter and P. Paigi, "Microgrid: A conceptual solution," in *Proc. 35th IEEE Annu. Power Electron. Spec. Conf. (PESC)*, Aachen, Germany, 2004, pp. 4285–4290.

[5] L. Che, M. Shahidehpour, and A. S. Alabdulwahab and Y. Al-Turki, "Hierarchical coordination of a community microgrid with AC and DC microgrids," *IEEE Trans. Smart Grid*, vol. 6, no. 6, pp. 3042–3051, Nov. 2015.

[6] F. Li, B. Luo, and P. Liu, "Secure information aggregation for smart grids using homomorphic encryption," in *Proc. 1st IEEE Int. Conf. Smart Grid Commun. (SmartGridComm)*, Gaithersburg, MD, USA, 2010, pp. 327–332.

[7] 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 Trans. Parallel Distrib. Syst.*, vol. 23, no. 9, pp. 1621–1631, Sep. 2012.

[8] L. E. Zubieta, "Power management and optimization concept for DC microgrids," in *Proc. 1st IEEE Int. Conf. DC Microgrids (ICDCM)*, Atlanta, GA, USA, 2015, pp. 81–85.

[9] M. Lisovich and S. Wicker, "Privacy concerns in upcoming residential and commercial demand-response systems," *IEEE Proc. Power Syst.*, vol. 1, no. 1, pp. 1–10, 2008.

[10] C. Peng, H. Sun, M. Yang, and Y.-L. Wang, "A survey on security communication and control for smart grids under malicious cyber attacks," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 49, no. 8, pp. 1554–1569, Aug. 2019.

[11] S. M. Amin and B. F. Wollenberg, "Toward a smart grid: Power delivery for the 21st century," *IEEE Power Energy Mag.*, vol. 3, no. 5, pp. 34–41, Sep./Oct. 2005.

[12] S. Li, K. Xue, Q. Yang, and P. Hong, "PPMA: Privacy-preserving multisubset data aggregation in smart grid," *IEEE Trans. Ind. Informat.*, vol. 14, no. 2, pp. 462–471, Feb. 2018.

[13] K. Xue, B. Zhu, Q. Yang, D. S. L. Wei, and M. Guizani, "An efficient and robust data aggregation scheme without a trusted authority for smart grid," *IEEE Internet Things J.*, vol. 7, no. 3, pp. 1949–1959, Mar. 2020.

[14] M. Ghadamyari and S. Samet, "Privacy-preserving statistical analysis of health data using paillier homomorphic encryption and permissioned blockchain," in *Proc. IEEE Int. Conf. Big Data (BigData)*, Los Angeles, CA, USA, 2019, pp. 5474–5479.

[15] B.-K. Zheng *et al.*, "Scalable and privacy-preserving data sharing based on blockchain," *J. Comput. Sci. Technol.*, vol. 33, no. 3, pp. 557–567, 2018.

[16] A. Bagherian and S. M. M. Tafreshi, "A developed energy management system for a microgrid in the competitive electricity market," in *Proc. IEEE Bucharest PowerTech*, Bucharest, Romania, 2009, pp. 1–6.

[17] H. Kakigano, Y. Miura, and T. Ise, "Distribution voltage control for DC microgrids using fuzzy control and gain-scheduling technique," *IEEE Trans. Power Electron.*, vol. 28, no. 5, pp. 2246–2258, May 2013.

[18] A. Khorsandi, M. Ashourloo, and H. Mokhtari, "A decentralized control method for a low-voltage DC microgrid," *IEEE Trans. Energy Convers.*, vol. 29, no. 4, pp. 793–801, Dec. 2014.

[19] L. Xue, Y. Teng, Z. Zhang, J. Li, K. Wang, and Q. Huang, "Blockchain technology for electricity market in microgrid," in *Proc. 2nd Int. Conf. Power Renew. Energy (ICPRE)*, Chengdu, China, 2017, pp. 704–708.

[20] E. Mengelkamp, B. Notheisen, C. Beer, D. Dauer, and C. Weinhardt, "A blockchain-based smart grid: Towards sustainable local energy markets," *Comput. Sci. Res. Develop.*, vol. 33, no. 1, pp. 207–214, 2018.

[21] Z. Li, J. Kang, R. Yu, D. Ye, Q. Deng, and Y. Zhang, "Consortium blockchain for secure energy trading in industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 14, no. 8, pp. 3690–3700, Aug. 2018.

[22] P. Danzi, M. Angjelichinoski, C. Stefanovic, and P. Popovski, "Distributed proportional-fairness control in microgrids via blockchain smart contracts," in *Proc. IEEE Int. Conf. Smart Grid Commun. (SmartGridComm)*, Dresden, Germany, 2017, pp. 45–51.

[23] S. Nakamoto. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. Accessed: Jun. 2021. [Online]. Available: https://www.bitcoinpaper.info/bitcoinpaper-html/

[24] V. Buterin *et al.*, "A next generation smart contract and decentralized application platform," Ethereum, Zug, Switzerland, White Paper, 2014. Accessed: Jun. 2021. [Online]. Available: https://cryptorating.eu/whitepapers/Ethereum/Ethereum_white_paper.pdf

[25] *Solidity*. Accessed: Jun. 2021. [Online]. Available: https://docs.soliditylang.org

[26] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in *Proc. USENIX Symp. Oper. Syst. Design Implement. (OSDI)*, 1999, pp. 173–186.

[27] P. Paillier, "Paillier encryption and signature schemes," in *Encyclopedia of Cryptography and Security*, H. C. A. V. Tilborg and S. Jajodia, Eds. Boston, MA, USA: Springer, 2011. [Online]. Available: https://doi.org/10.1007/978-1-4419-5906-5_488

[28] F. Marini and B. Walczak, "Particle swarm optimization (PSO). A tutorial," *Chemometr. Intell. Lab. Syst.*, vol. 149, pp. 153–165, Dec. 2015.

Xinyi Luo (Graduate Student Member, IEEE) received the B.S. degree in information security from the School of the Gifted Young, University of Science and Technology of China in July, 2020, where she is currently a graduated student with the School of Cyber Security. Her research interests include network security and cryptography.

Kaiping Xue (Senior Member, IEEE) received the bachelor's degree from the Department of Information Security, University of Science and Technology of China (USTC), in 2003, and the Ph.D. degree from the Department of Electronic Engineering and Information Science (EEIS), USTC, in 2007. From May 2012 to May 2013, he was a Postdoctoral Researcher with the Department of Electrical and Computer Engineering, University of Florida. He is currently a Professor with the School of Cyber Security and the Department of EEIS, USTC. His research interests include next-generation Internet architecture design, transmission optimization, and network security. He serves on the Editorial Board of several journals, including the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, and the IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT. He has also served as a Guest Editor of IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, and a Lead Guest Editor of *IEEE Communications Magazine* and IEEE NETWORK. He is an IET Fellow.

Jie Xu received the B.S. degree from the Department of Information Security, University of Science and Technology of China (USTC) in July, 2017, and the M.S. degree from the Department of Electronic Engineering and Information Science, USTC, in 2020. She is currently pursuing the Ph.D. degree with the Department of Computer Science, City University of Hong Kong. Her research interests include network security and cryptography.

Qibin Sun (Fellow, IEEE) received the Ph.D. degree from the Department of Electronic Engineering and Information Science, University of Science and Technology of China, in 1997, where he is currently a Professor with the School of Cyber Security. He has published more than 120 papers in international journals and conferences. His research interests include multimedia security, network intelligence, and security.

Yongdong Zhang (Senior Member, IEEE) received the Ph.D. degree in electronic engineering from Tianjin University, Tianjin, China, in 2002. He is currently a Professor with the Department of Electronic Engineering and Information Science and the Beijing Research Institute, University of Science and Technology of China. He has authored over 100 refereed journal and conference papers. His current research interests are in the fields of multimedia content analysis and understanding, multimedia content security, video encoding, and streaming media technology. He was a recipient of the Best Paper Awards in PCM 2013, ICIMCS 2013, and ICME 2010, and the Best Paper Candidate at ICME 2011. He is a member of the Editorial Board of the IEEE TRANSACTIONS ON MULTIMEDIA and *Multimedia Systems*.