Blockchain-Enabled Decentralized Privacy-Preserving Group Purchasing for Retail Energy Plans

Sid Chi-Kin Chau
sid.chau@anu.edu.au
Australian National University

Yue Zhou
yue.zhou@anu.edu.au
Australian National University

ABSTRACT

Retail energy markets are increasingly consumer-oriented, thanks to a growing number of energy plans offered by a plethora of energy suppliers, retailers and intermediaries. To maximize the benefits of competitive retail energy markets, *group purchasing* is an emerging paradigm that aggregates consumers' purchasing power by coordinating switch decisions to specific energy providers for discounted energy plans. Traditionally, group purchasing is mediated by a trusted third-party, which suffers from the lack of privacy and transparency. In this paper, we introduce a novel paradigm of *decentralized privacy-preserving* group purchasing, empowered by privacy-preserving blockchain and secure multi-party computation, to enable users to form a coalition for coordinated switch decisions in a decentralized manner, without a trusted third-party. The coordinated switch decisions are determined by a competitive online algorithm, based on users' private consumption data and current energy plan tariffs. Remarkably, no private user consumption data will be revealed to others in the online decision-making process, which is carried out in a transparently verifiable manner to eliminate frauds from dishonest users and supports fair mutual compensations by sharing the switching costs to incentivize group purchasing. We implemented our decentralized group purchasing solution as a smart contract on Solidity-supported blockchain platform (e.g., Ethereum), and provide extensive empirical evaluation.

CCS CONCEPTS

• **Security and privacy** → **Privacy-preserving protocols**; **Privacy protections**; **Security protocols**;

KEYWORDS

Privacy-Preserving Blockchain, Group Purchasing, Energy Plans, Competitive Online Algorithm, Secure Multi-party Computation

1 INTRODUCTION

With rising deregulation and decarbonization in the traditionally monopolized energy sector, a myriad of energy plans and tariffs are being offered in retail energy markets in the US, Europe, and other countries [40]. In today's competitive retail energy markets with increasing choices, users are able to compare and switch among diverse energy plans from multiple retail energy providers [38], with different tariff structures (e.g., hourly rates, peak/off-peak hours, PV feed-in tariffs) and contractual arrangements (e.g., connection/disconnection fees, contracted periods). Some energy plans offer special subsidies (e.g., incentive packages for home batteries and energy-efficient boilers), and options for renewable energy and carbon offsetting. With a growing set of energy suppliers and retailers, the number of energy plans has mushroomed significantly. For instance, 160+ energy plans are available in Buffalo, the US [22], while 400+ electricity plans are available in Sydney, Australia [36]. This presents ample opportunities for end users to cherry-pick the best plans that optimize their energy bills and needs.

Along with a variety of energy plans, a new consumer-driven paradigm called *group purchasing* (or bulk buying) has emerged in retail energy markets [41], which boosts consumers' purchasing power by coordinated purchasing decisions in a coalition of consumers. There has been a long history of group purchasing in various sectors [13]. The idea that consumers should aggregate their demands to increase bargaining power with vendors has been practiced in health care and e-commerce (e.g., Groupons, Meituan). Recently, there emerged several group purchasing start-ups in the energy sector that operate as intermediaries between energy suppliers and users. Some of them recruit users for discounted group-based energy plans, whereas others solicit group-based tenders from energy suppliers on behalf of their users (e.g., ChoiceEnergy [15]). With group purchasing, users can collectively switch to specific energy suppliers for better discounts by leveraging their aggregate purchasing power. Hence, group purchasing for energy plans is becoming a popular paradigm in retail energy markets.

Traditionally, group purchasing is mediated by a trusted third-party agent in a centralized manner, such that users are required to submit their personal profiles and private data (e.g., past energy bills) to an agent, who will negotiate with some vendors on their behalf. In exchange for discounted energy plans for users, the agent will receive commission fees from the vendors. However, there is a lack of *transparency* in the third-party mediation approach. The agent may obscure the commission fees and negotiation process, and may not maximize the users' interest, but rather the interest of the agent or the vendors. Moreover, there is a *privacy* concern – users may be unaware of the potential misuses and breaches of their private data by the agent for other unintended purposes. To bolster user privacy, stricter privacy protection legislations (e.g., GDPR in Europe) are introduced in various countries to restrict personal information access by a third-party. Yet, private user data can still be exploited through other illicit approaches (e.g., hacking into the agent). Therefore, there is a need to ensure both transparency and privacy in the decision-making processes of group purchasing, while enjoying its benefits.

e-Energy '22, June 28–July 1, 2022, Virtual Event

ACM ISBN 978-1-4503-9397-3/22/06…$15.00
https://doi.org/10.1145/3538637.3538848

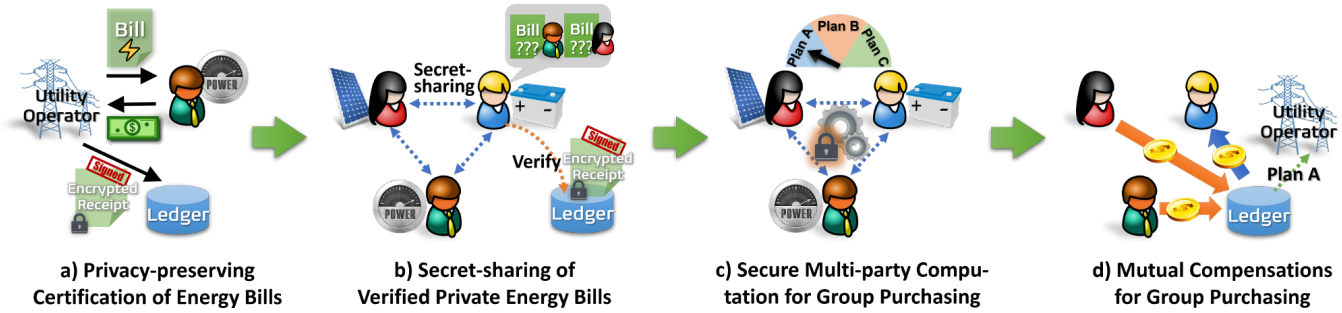

Figure 1: An illustration of our solution for decentralized privacy-preserving group purchasing.

In this paper, we introduce a novel paradigm of "*decentralized privacy-preserving group purchasing*", which removes the trusted third-party intermediaries. Our goal is to enable users to form a coalition for making coordinated energy plan switch decisions in a decentralized manner with assurance of privacy and transparency:

(1) **Privacy Protection of User Data:** Deciding a suitable energy plan requires a sophisticated consideration of a variety of factors, such as past consumption data and current energy plan tariffs. However, sharing personal information with others in group purchasing may be undesirable. We need to guarantee that no private user consumption data will be revealed to other parties for unintended purposes, while ensuring a discreet decision-making process that properly incorporates the data of all users in group purchasing.

(2) **Transparently Verifiable Decision-Making:** Without the knowledge of its private input, privacy poses a significant challenge to the verifiability of the decision-making process. In particular, dishonest users may take advantage of privacy protection to cheat or misrepresent their energy data. These dishonest users are likely to collude to coordinate their actions. Hence, it is critical to safeguard against the potential presence of a large number of dishonest users. We need to ensure that the decision-making process should be transparent and verifiable to eliminate any fraud from dishonest users, while preserving user privacy.

In this paper, we propose an effective solution to support decentralized privacy-preserving group purchasing. Our solution draws on several components: (1) a competitive online algorithm for group purchasing decision-making, (2) secure multi-party computation for privacy-preserving online decision-making, and (3) zero-knowledge proofs on blockchain for validating the private input data to our online algorithm. We briefly explain these components as follows, but the details will be elaborated in the subsequent sections.

Competitive Online Algorithm: First, the problem of energy plan selection [38] without future knowledge (e.g., future energy demands and tariffs of future energy plans) belongs to *online decision-making problems*. There is an extensive body of literature [7] on online algorithms that solve these online problems with theoretical-proven bounds on the optimality of their online decisions (as known as competitive ratios). In this paper, we formulate an online decision problem of group purchasing for energy plan as a Metrical Task System problem, and devise a competitive online algorithm that produces close-to-optimal performance in our evaluation. Our online algorithm extends the related work [41]. Furthermore, our online algorithm supports *mutual compensations* in group purchasing, such that some users may be fairly compensated by others for the switching costs to join group purchasing. Mutual compensations effectively incentivize group purchasing.

Secure Multi-party Computation: Second, we execute our competitive online algorithm for group purchasing in a privacy-preserving manner, without disclosing the private input data (e.g., users' energy bills). We rely on secure multi-party computation (or simply called multi-party computation (MPC)), which is a general framework to compute a function jointly by multiple parties with concealed input from each other. Recently, SPDZ [17], an efficient MPC protocol based on secret-sharing, has been applied to many practical applications (e.g., machine learning [14]). SPDZ can safeguard against a dishonest majority of users ($\geq$ 50% of users may be dishonest). In this paper, we apply SPDZ to the online decision-making process of group purchasing with concealed input data.

Blockchain and Zero-knowledge Proofs: Third, it is important to validate the private input data, before the online decision-making process. Otherwise, dishonest users may input falsified data to influence the outcomes for their benefits. We utilize blockchain to record users' energy bills, and employ smart contracts for verifying mutual compensations among users. There are several functions of blockchain. (i) Blockchain is a decentralized platform for verifiable applications without trusted intermediaries, which enables decentralized verification of mutual compensations to match the decision of the online algorithm. (ii) The crypto-tokens on blockchain serve as a convenient micro-payment system for compensations among users. (iii) Blockchain is a public non-temperable data repository for energy bill data. However, blockchain by default does not protect privacy, and the data is disclosed publicly on the ledger. Therefore, we apply zero-knowledge proofs (ZKPs), which can verify certain properties of encrypted data, without decrypting it. For example, instead of storing users' energy bills, we record the corresponding encrypted receipts on blockchain. Then users can use ZKPs to validate the input data for online decision-making with respect to the encrypted receipts, without disclosing the original data.

Together with these components, we develop an effective solution to support decentralized privacy-preserving group purchasing, as illustrated at a high level in Figure 1:

a) First, the utility operator records the encrypted receipts of energy bills on the blockchain ledger, after users pay their bills. But no private energy bills are revealed on the ledger.
b) The users use multi-party computation (SPDZ) to share their energy bill data in a privacy-preserving manner, without disclosing the private data to each other. They will jointly verify the validity of private data, via SPDZ, based on ZKPs.
c) With verified private input data, the users jointly execute the online algorithm via SPDZ to decide the suitable group purchasing decision that maximizes the benefits of all users, if possible. Moreover, they will also decide mutual compensations to incentivize each other to join group purchasing.
d) The group purchasing decision and mutual compensations are handled and recorded on the blockchain. The utility operator executes the chosen energy plan according to the records on the blockchain. The users are assured of transparency in the whole decision-making process.

The technical details of the procedures will be presented in the rest of the paper. We remark that although our solution is designed for energy plan selection, it is rather generic and generalizable to other group purchasing applications (e.g., health care plans).

This paper is organized as follows. We first formulate a decision-making model, in which users form a coalition for group purchasing of energy plans without the knowledge of future energy demands and tariffs. In particular, we consider mutual compensations in online decision-making to incentivize group purchasing. We next formulate the security model considering privacy protection and possible attacks from dishonest users. We then present the basics of cryptographic components and multi-party computation. The privacy-preserving solution is presented with an evaluation of our implementation on Solidity-supported blockchain platform.

2 DECISION-MAKING MODEL

In the following, we formulate a problem of group purchasing for energy plans as an online decision-making problem that is possibly conducted in a centralized manner, without considering privacy. In the subsequent sections, we will incorporate privacy protection in a decentralized mechanism. First, we will consider the standalone setting of energy plan selection without group purchasing. Then we will extend our study to the setting with group purchasing. Finally, we will present a group purchasing decision-making mechanism.

2.1 Problem Setup

Our problem of energy plan selection is consisted of the following components:

- **Energy Plans:** We consider discrete timeslots, indexed by $t \in \{1, ..., T\}$. There are a set of energy plans $\mathcal{E}$. Each energy plan is indexed by $j \in \mathcal{E}$ and characterized by a tuple $\mathsf{E}(j) = \langle \mathsf{p}_+^t(j), \mathsf{p}_-^t(j), \mathsf{c}(j), \mathsf{d}(j), \mathsf{T}(j) \rangle$, as described as follows:
  (1) *Consumption Tariffs*: Let $\mathsf{p}_+^t(j)$ be a pricing function that maps a timeslot t to its respective time-of-use price of energy consumption (i.e., energy import). $\mathsf{p}_+^t(j)$ can capture a different rate at timeslot t for the peak, off-peak and shoulder periods.
  (2) *Feed-in Tariffs*: We also consider the feed-in rates of energy production (i.e., energy export from PV or home batteries). Let $\mathsf{p}_-^t(j)$ be a pricing function that maps a timeslot t to its respective time-varying price of energy production.
  (3) *Connection Fee*: Let $\mathsf{c}(j)$ be the connection fee when a user joins energy plan j. Typical connection fee includes the set-up and installation costs at the distribution grid.
  (4) *Disconnection Fee*: Let $\mathsf{d}(j)$ be the disconnection fee when a user terminates energy plan j before the contract period. Higher disconnection fees can discourage users from early termination of the energy plans. For simplicity, we consider constant disconnection fees, regardless the residual period of the contract duration.
  (5) *Contract Duration*: Let $\mathsf{T}(j)$ be the contract duration, beyond which the energy plan can be terminated without incurring any disconnection fee.
- **Energy Users:** There are a set of users $\mathcal{N}$, indexed by $i \in \{1, ..., N\}$. Each user i has certain energy consumption or production rates over time, represented by a sequence (a_i^t), where $a_i^t > 0$ denotes energy consumption and $a_i^t < 0$ denotes energy production. Let $x_i^t \in \mathcal{E}$ be the selected energy plan by user i at timeslot t. Let $x_i = (x_i^1, ..., x_i^T)$ be a sequence of user i's selections over time, and its feasible space be $\mathcal{E}^T$. Let the long-term energy cost incurred by selections x_i be

$$\mathsf{Cost}_i[x_i] \triangleq \sum_{t=1}^{T} \left(\mathsf{Cost}_{\mathsf{Op}}^t[x_i^t, a_i^t] + \mathsf{Cost}_{\mathsf{Sw}}^t[x_i^{t-1}, x_i^t] \right) \quad (1)$$

where $\mathsf{Cost}_{\mathsf{Op}}^t[x_i^t, a_i^t]$ is the operational cost defined by

$$\mathsf{Cost}_{\mathsf{Op}}^t[x_i^t, a_i^t] \triangleq \mathsf{p}_+^t(x_i^t) \cdot a_i^t \cdot \mathbb{1}\{a_i^t \geq 0\} + \mathsf{p}_-^t(x_i^t) \cdot a_i^t \cdot \mathbb{1}\{a_i^t < 0\}$$

and $\mathsf{Cost}_{\mathsf{Sw}}^t[x_i^{t-1}, x_i^t]$ is the switching cost between energy plans defined by

$$\mathsf{Cost}_{\mathsf{Sw}}^t[x_i^{t-1}, x_i^t] \triangleq \left(\mathsf{c}(x_i^t) + \mathsf{d}(x_i^{t-1}) \cdot \mathbb{1}\{\mathsf{T}_i^t \leq \mathsf{T}(x_i^{t-1})\} \right) \cdot \mathbb{1}\{x_i^t \neq x_i^{t-1}\}$$

Let T_i^t be the number of previous consecutive timeslots that user i has been using energy plan x_i^{t-1} before timeslot t.
The goal of each user i is to decide x_i, as to minimize her long-term total energy cost in the following problem:

$$(\mathsf{P}_i) \qquad \min_{x_i \in \mathcal{E}^T} \mathsf{Cost}_i[x_i] \quad (2)$$

However, each user can only do so in an online manner without the future knowledge of $a_i^{t'}$, $\mathsf{p}_+^{t'}(j)$ and $\mathsf{p}_-^{t'}(j)$ for any j and $t' > t$ at each timeslot t, because there may be uncertain future demands and tariffs, or new energy plans.

2.2 Standalone Online Energy Plan Selection

In this section, we solve (P_i) in the standalone setting without group purchasing. This problem has been studied previously in [38]. In this subsection, we present some preliminaries for this problem.

Problem (P_i) belongs to a general problem class called *Metrical Task System* (MTS) problem [7], which consists of a set of states (e.g., energy plans). Each state is associated with a time-varying cost, and there is a switching cost if the decision-maker switches from one state to another. The goal of the decision-maker is to decide a sequence of states to minimize the total of state costs and switching costs, without knowing the future state costs. The

MTS problem captures a wide range of scenarios, such as k-server, energy generation dispatching, and energy plan selection problems.

We describe the offline and online algorithms for solving (P_i):

(1) **Offline Algorithm:** The offline optimal solution of (P_i) can be computed by dynamic programming. Formally, let $\mathtt{Opt}_i^t[x_i^t]$ be the optimal cost of user i with selection $x_i^t \in \mathcal{E}$ at timeslot t. When $\mathtt{Opt}_i^{t-1}[x]$ is known for all $x \in \mathcal{E}$, $\mathtt{Opt}_i^t[x_i^t]$ can be computed iteratively by Bellman-Ford equation:

$$\mathtt{Opt}_i^t[x_i^t] = \min_{x\in\mathcal{E}} \left(\mathtt{Opt}_i^{t-1}[x] + \mathtt{Cost}_{\mathsf{Op}}^t[x_i^t, a_i^t] + \mathtt{Cost}_{\mathsf{Sw}}^t[x, x_i^t]\right)$$

Initially, we set $\mathtt{Opt}_i^0[x] = 0$ for all x. Then, we can compute $\mathtt{Opt}_i^t[x]$ for all x using $\mathtt{Opt}_i^{t-1}[x]$ by dynamic programming. Let $(x^{*t}_i)_{t=1}^T$ be an offline optimal selection of (P_i), which is obtained by backward computation. First, we obtain x^{*T}_i by

$$x^{*T}_i = \arg\min_{x\in\mathcal{E}} \mathtt{Opt}_i^T[x] \tag{3}$$

Then, we can obtain $x^{*t-1}_i = x$ using x^{*t}_i by finding a suitable x that satisfies

$$\mathtt{Opt}_i^t[x^{*t}_i] = \mathtt{Opt}_i^{t-1}[x] + \mathtt{Cost}_{\mathsf{Op}}^t[x^{*t}_i, a_i^t] + \mathtt{Cost}_{\mathsf{Sw}}^t[x, x^{*t}_i] \tag{4}$$

(2) **Online Algorithm:** To make online decisions for (P_i), one can only rely on the present or past knowledge at timeslot t, without the future knowledge of $a_i^{t'}$, $\mathsf{p}_+^{t'}(j)$ and $\mathsf{p}_-^{t'}(j)$ for any j and $t' > t$. Deciding an online decision is a hard problem, because of the presence of switching costs. Switching to a state with low current state cost may not offset the switching cost, when the state cost increases in the future. An online algorithm for (P_i) is known as *Work Function algorithm* (WFA) [7]. The basic idea of WFA is to find a selection that minimizes the discrepancy with the offline optimal cost at the current timeslot t, subject to the constraint that the selection does not switch from the previous timeslot in the offline optimal cost. Formally, let $\hat{x}_i^t$ be the selection produced by WFA at timeslot t. $\hat{x}_i^t$ is decided by the following equation:

$$\hat{x}_i^t = \arg\min_{x\in\mathcal{E}} \left(\mathtt{Opt}_i^t[x] + \mathtt{Cost}_{\mathsf{Sw}}^t[\hat{x}_i^{t-1}, x]\right) \tag{5}$$

subject to

$$\mathtt{Opt}_i^t[x] = \mathtt{Opt}_i^{t-1}[x] + \mathtt{Cost}_{\mathsf{Op}}^t[x, a_i^t] \tag{6}$$

where $\hat{x}_i^{t-1}$ is the selection of the previous timeslot by WFA. Constraint (6) means that only the selections without switching at the previous timeslot in the offline optimal cost are the feasible candidates. Note that there always exists $x \in \mathcal{E}$ that satisfies Constraint (6), namely, at least one x that does not switch from the previous timeslot in the offline optimal solution. Otherwise, $\mathtt{Opt}_i^t[x]$ is not optimal. We remark that the computation of $\mathtt{Opt}_i^t[x]$ does not need any knowledge of future timeslot $t' > t$. Hence, WFA is an online algorithm.

Following the standard terminology of competitive analysis [7], we define the *competitive ratio* of an online algorithm by the worst-case ratio between the online solution and offline optimal solution: $\max \frac{\mathtt{Cost}_i[\hat{x}_i]}{\mathtt{Opt}_i^T[x^{*T}_i]}$ over all possible inputs. There is a general upper bound of the competitive ratio of WFA.

Theorem 1 ([7]). *If there are n states in a MTS problem, then the competitive ratio of WFA is upper bounded by $2n-1$. Namely, in our problem, each state is an energy plan, and $n = |\mathcal{E}|$. Hence,*

$$\mathtt{Cost}_i[\hat{x}_i] \le (2|\mathcal{E}| - 1) \cdot \mathtt{Opt}_i^T[x^{*T}_i]$$

If there are only two energy plans, then we obtain $\mathtt{Cost}_i[\hat{x}_i] \le 3 \cdot \mathtt{Opt}_i^T[x^{*T}_i]$. In fact, we can show a constant upper bound on the competitive ratio of WFA in a special case.

Theorem 2. *If the connection fees and disconnection fees of all energy plans are identical, i.e., $\mathsf{c}_j = \mathsf{c}$ and $\mathsf{d}_j = \mathsf{d}$ for all $j \in \mathcal{E}$, then*

$$\mathtt{Cost}_i[\hat{x}_i] \le 3 \cdot \mathtt{Opt}_i^T[x^{*T}_i]$$

Remarkably, even though the (dis)connection fees are not exactly identical, it is observed that the empirical competitive ratio is still well below the upper bound 3 in our evaluation.

2.3 Group-based Online Energy Plan Selection

In this section, we incorporate group purchasing into our energy selection problem. We consider a group-based energy plan denoted by $\mathfrak{g}$, in addition to other individual energy plans $\mathcal{E}\backslash\{\mathfrak{g}\}$. Group-based energy plan $\mathfrak{g}$ usually has more favorable tariffs. But there are other characteristics of a group-based energy plan in practice, such as the requirement of a minimum number of sign-up users:

(1) ***Joining***: To join energy plan $\mathfrak{g}$, there requires a minimum number of $\mathsf{N}(\mathfrak{g})$ sign-up users. One may interpret $\mathsf{N}(\mathfrak{g})$ as the minimum aggregate bargaining power. When we set $\mathsf{N}(\mathfrak{g}) = 1$, then any user can join $\mathfrak{g}$, without group purchasing.
(2) ***Termination***: We consider a simple setting of termination. Any user can unilaterally terminate plan $\mathfrak{g}$ before the contract period by paying the disconnection fee $\mathsf{d}(\mathfrak{g})$, which may be higher than any individual energy plan.

We next extend the online algorithm WFA to the setting of group purchasing for energy plans. Suppose $\mathfrak{g} \in \mathcal{E}$ is the only group-based plan (i.e., $\mathsf{N}(\mathfrak{g}) > 1$), while the rest are individual energy plans (i.e., $\mathsf{N}(j) = 1$, for all $j \in \mathcal{E}\backslash\{\mathfrak{g}\}$). Given a subset of users $X \subseteq \mathcal{N}$ (who have not joined group-based plan $\mathfrak{g}$), we will decide whether X should join group energy plan $\mathfrak{g}$ or not. If so, how to meet the requirement of minimum number of users when sign-up.

Let $(\hat{x}_i^{t-1})_{i\in X}$ be the users' selections of the previous timeslot $t-1$. Naturally, each user $i \in X$ applies WFA to the two following sub-problems:

(1) Considering only group-based plan $\mathfrak{g}$: If $\mathtt{Opt}_i^t[\mathfrak{g}] \neq \mathtt{Opt}_i^{t-1}[\mathfrak{g}] + \mathtt{Cost}_{\mathsf{Op}}^t[g, a_i^t]$ for any user $i \in X$ (namely, Constraint (6) is violated), then $\mathfrak{g}$ will not be considered. Otherwise, let

$$C_i^t(\mathfrak{g}) \triangleq \mathtt{Opt}_i^t[\mathfrak{g}] + \mathtt{Cost}_{\mathsf{Sw}}^t[\hat{x}_i^{t-1}, g]$$

(2) Considering the rest of energy plans $\mathcal{E}\backslash\{\mathfrak{g}\}$: Find $y_i \in \mathcal{E}\backslash\{\mathfrak{g}\}$, such that

$$y_i = \arg\min_{y\in\mathcal{E}\backslash\{\mathfrak{g}\}} \left(\mathtt{Opt}_i^t[y] + \mathtt{Cost}_{\mathsf{Sw}}^t[\hat{x}_i^{t-1}, y]\right)$$

subject to $\mathtt{Opt}_i^t[y] = \mathtt{Opt}_i^{t-1}[y] + \mathtt{Cost}_{\mathsf{Op}}^t[y, a_i^t]$. Let

$$C_i^t(\mathcal{E}\backslash\{\mathfrak{g}\}) \triangleq \mathtt{Opt}_i^t[y_i] + \mathtt{Cost}_{\mathsf{Sw}}^t[\hat{x}_i^{t-1}, y_i]$$

Let $C_i^t(\mathcal{E}\backslash\{\mathfrak{g}\}) = \infty$, if no y satisfies Constraint (6).

Let us first consider the setting where each user decides separately whether to join group-based plan $\mathfrak{g}$. Let $m_X^t(\mathfrak{g})$ be the number of users in X who strictly prefer $\mathfrak{g}$, namely,

$$m_X^t(\mathfrak{g}) \triangleq \left|\left\{i \in X \mid C_i^t(\mathfrak{g}) \le C_i^t(\mathcal{E}\backslash\{\mathfrak{g}\})\right\}\right|$$

If $m_X^t(\mathfrak{g}) \ge \mathsf{N}(\mathfrak{g})$, then these $m_X^t(\mathfrak{g})$ users will naturally join energy plan $\mathfrak{g}$. However, if $m_X^t(\mathfrak{g}) < \mathsf{N}(\mathfrak{g})$, then these users are unable to join energy plan $\mathfrak{g}$, without the required minimum number of sign-up users. In the next section, we will incorporate *mutual compensations* to incentivize users to join energy plan $\mathfrak{g}$ by compensating their switching costs of terminating their current energy plans, when $m_X^t(\mathfrak{g}) < \mathsf{N}(\mathfrak{g})$.

2.4 Mutual Compensations for Group Purchasing

In this section, we consider the possibility that some users will pay mutual compensations to other users for partially subsidizing their switching costs in order to join group-based plan $\mathfrak{g}$.

It is not always true that $C_i^t(\mathfrak{g}) \le C_i^t(\mathcal{E}\backslash\{\mathfrak{g}\})$ for all $i \in X$, as some users may have $C_i^t(\mathcal{E}\backslash\{\mathfrak{g}\}) < C_i^t(\mathfrak{g})$. However, if $\sum_{i\in X} C_i^t(\mathfrak{g}) \le \sum_{i\in X} C_i^t(\mathcal{E}\backslash\{\mathfrak{g}\})$ (i.e., the sum, instead of individual users), then some users may compensate others in order to join energy plan $\mathfrak{g}$. Let θ_i be the *compensated switching cost* for user i. Hence, rather than considering $C_i^t(\mathfrak{g})$ in WFA, each user i considers the following compensated cost:

$$\mathtt{Opt}_i^t[\mathfrak{g}] + \theta_i \tag{7}$$

Note that some users may pay more (i.e., $\theta_i > \mathtt{Cost}_{\mathsf{Sw}}^t[\hat{x}_i^{t-1}, g]$), while other users may pay less (i.e., $\theta_i < \mathtt{Cost}_{\mathsf{Sw}}^t[\hat{x}_i^{t-1}, g]$).

To find proper values for $(\theta_i)_{i\in X}$, we consider several properties:

- ***Individual Rationality***: With compensated switching cost, each user can apply WFA individually to decide whether to join group-based plan $\mathfrak{g}$ in an online manner. All users prefer group-based plan $\mathfrak{g}$, if the following condition of *individual rationality* is satisfied for all $i \in X$:

$$\mathtt{Opt}_i^t[\mathfrak{g}] + \theta_i < C_i^t(\mathcal{E}\backslash\{\mathfrak{g}\}) \tag{8}$$

- ***Budget Balance***: We consider no external financial support, namely, the total compensated switching cost should equal the default switching cost without mutual compensations. Hence, the following condition of *budget balance* is required:

$$\sum_{i\in X} \theta_i = \sum_{i\in X} \mathtt{Cost}_{\mathsf{Sw}}^t[\hat{x}_i^{t-1}, g] \tag{9}$$

Equivalently, $\sum_{i\in X}\left(\mathtt{Opt}_i^t[\mathfrak{g}] + \theta_i\right) = \sum_{i\in X} C_i^t(\mathfrak{g})$.

- ***Group Feasibility***: From the perspective of a social planner, we also consider the following social online decision problem with aggregate cost of all users, as defined as follows:

$$(\mathrm{P_{soc}}) \qquad \min_{x\in\mathcal{E}^{T|X|}} \sum_{i\in X} \mathtt{Cost}_i[x_i] \tag{10}$$

where $x = (x_i)_{i\in X}$ is the collective selections of all users, and $\mathcal{E}^{T|X|}$ is the feasible space. Unlike the individual problem (P_i), the social problem $(\mathrm{P_{soc}})$ minimizes the total long-term total energy cost of all users in X. When we apply WFA to problem $(\mathrm{P_{soc}})$, the online decision to switch to group-based plan $\mathfrak{g}$ is feasible, if the following condition of *group feasibility* is satisfied:

$$\sum_{i\in X} C_i^t(\mathfrak{g}) < \sum_{i\in X} C_i^t(\mathcal{E}\backslash\{\mathfrak{g}\}) \tag{11}$$

In practice, individual rationality and budget balance are important properties of a feasible scheme of mutual compensations, whereas group feasibility can be easily checked before deciding mutual compensations.

Theorem 3. *We consider two schemes of mutual compensations (based on the ideas in [10–12]):*

(1) **Egalitarian Cost-sharing:**

$$\theta_i = C_i^t(\mathcal{E}\backslash\{\mathfrak{g}\}) + \frac{\sum_{i'\in X}\left(C_{i'}^t(\mathfrak{g}) - C_{i'}^t(\mathcal{E}\backslash\{\mathfrak{g}\})\right)}{|X|} - \mathtt{Opt}_i^t[\mathfrak{g}] \tag{12}$$

(2) **Proportional Cost-sharing:**

$$\theta_i = C_i^t(\mathcal{E}\backslash\{\mathfrak{g}\})\frac{\sum_{i'\in X} C_{i'}^t(\mathfrak{g})}{\sum_{i'\in X} C_{i'}^t(\mathcal{E}\backslash\{\mathfrak{g}\})} - \mathtt{Opt}_i^t[\mathfrak{g}] \tag{13}$$

Then $(\theta_i)_{i\in X}$ in both schemes satisfy individual rationality and budget balance, when group feasibility is satisfied.

See Appendix A for the proof. Theorem 3 provides two feasible schemes of mutual compensations that guarantee individual rationality and budget balance, given group feasibility. Hence, it suffices to consider the social problem $(\mathrm{P_{soc}})$ and apply WFA to $(\mathrm{P_{soc}})$ with respect to X.

2.5 Group Purchasing Decision-Making Mechanism

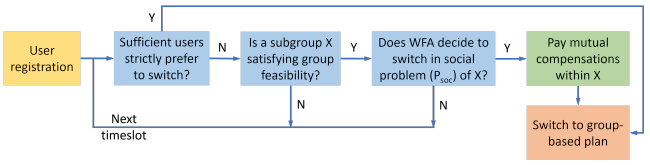

Figure 2: A flowchart of the group purchasing decision-making mechanism.

Based on the above results, this section presents a decision-making mechanism for group purchasing, incorporating mutual compensations. As illustrated in Figure 2, we determine the group-purchasing decision as follows:

Group Purchasing Decision-Making Mechanism:

(1) The users who are interested in joining group-based plan $\mathfrak{g}$ first register as $\mathcal{N}$. They agree on whether egalitarian cost-sharing or proportional cost-sharing will be used.
(2) At each timeslot t, check if there are sufficient users who strictly prefer to switch to group-based plan $\mathfrak{g}$ without mutual compensations. Namely, if $m_X^t(\mathfrak{g}) \ge \mathsf{N}(\mathfrak{g})$, then these $m_X^t(\mathfrak{g})$ users will join group-based plan $\mathfrak{g}$.
(3) Otherwise, if $m_X^t(\mathfrak{g}) < \mathsf{N}(\mathfrak{g})$
  (a) Find a subset $X \subseteq \mathcal{N}$, such that $|X| \ge \mathsf{N}(\mathfrak{g})$ and group feasibility is satisfied within X, namely, $\sum_{i\in X} C_i^t(\mathfrak{g}) < \sum_{i\in X} C_i^t(\mathcal{E}\backslash\{\mathfrak{g}\})$. Apply WFA on $(\mathrm{P_{soc}})$ with respect to X.

(b) If WFA decides to switch to group-based plan $\mathfrak{g}$ in $(\mathsf{P}_{\text{soc}})$ with X, then compute $(\theta_i)_{i\in X}$ of the respective mutual compensation scheme. Based on Theorem 3, individual rationality and budget balance are guaranteed.
(c) After paying the respective mutual compensations to each other, the users in X will join group-based plan $\mathfrak{g}$.

The above decision-making mechanism can be conducted in a centralized manner or via a third-party, when every user discloses their private energy data. In the rest of the paper, we seek to perform the decision-making mechanism in a decentralized privacy-preserving manner, without disclosing private energy data.

3 SECURITY & THREAT MODEL

In this section, we formulate privacy protection in the group purchasing decision-making mechanism. Note that we do not consider privacy protection in the communication. We assume that secure, reliable and authenticated communications can be established among the parties, with no man-in-the-middle attack. Secure communications can be attained by suitable end-to-end security.

3.1 Security and Privacy Requirements

In this paper, we provide a privacy-preserving solution to enable the users to jointly conduct the decision-making mechanism in Section 2.5 to determine the group purchasing decisions and mutual compensations, given their demands and current energy plans. Our privacy-preserving solution achieves the following security and privacy requirements in the decision-making mechanism:

(1) **Private Demands:** No users will leak their past demands (a_i^t) to other users. However, users will be able to verify their past demands from an authorized source, without revealing their demands to each other.
(2) **Private Selections:** No users will leak their currently selected energy plans (x_i^t) to other users, if they do not need to share the same group-based energy plan.
(3) **Private Compensations:** No users will leak their mutual compensations received from other users (θ_i^t), because it may reveal their demands or energy plans.

Nonetheless, we assume that all energy plan information $\mathsf{E}(j) = \langle \mathsf{p}_+^t(j), \mathsf{p}_-^t(j), \mathsf{c}(j), \mathsf{d}(j), \mathsf{T}(j)\rangle$ is publicly known for all $j \in \mathcal{E}$.

3.2 Threat Model

Note that dishonest user may take advantage of privacy protection to perform the following attacks in the decision-making process:

(1) ***Misrepresentation:*** Dishonest users may try to misrepresent their demands or current energy plans in the decision-making process.
(2) ***Mis-computation:*** They may output incorrect or inconsistent values in any joint computation process (e.g., validation or generation of proofs).
(3) ***Incorrect Compensations:*** They may try to cheat by paying less compensation to others, or claim more compensation than what they ought to.

Dishonest users may also collude with each other to coordinate their actions. To facilitate our solution, we make some assumptions about the dishonest users:

(1) *Maximum number:* Up to $(N-2)$ dishonest users, who may be adaptive adversaries and can deliberately deviate from the protocols for prying into others' privacy or sabotage the protocols. Their disruptive actions can happen at any time during the protocols rather than before the protocols (as for static adversaries). In case of any dishonest actions being detected, our system will abort and notify all the users.
(2) *Identifiability:* Our solution ensures that any dishonest actions will be detected. However, our system is not required to identify individual dishonest users, as it is fundamentally impossible [5] to identify a dishonest user in multi-party computation with a majority of dishonest users. There are secure multi-party computation protocols [16] that can identify a dishonest user, but requiring a majority of honest users and considerable computational overhead. On the other hand, we may impose further measures to mitigate dishonesty. For example, requiring proper user authentication to prevent shilling. Or, we can require each user to pay a deposit in advance, which will be forfeited if any dishonesty is detected.

4 SYSTEM MODEL

In this section, we present the system models of blockchain and energy bill data that underlie our privacy-preserving solution.

4.1 Blockchain Model

We consider an account-based blockchain model like Ethereum (which is a general-purpose blockchain platform [34]), whereas Bitcoin operates with a different transaction-output model for cryptocurrency transactions only. Smart contracts are programmable code on a blockchain platform that provide customized computation tasks along with each transaction (e.g., transaction validation, data processing). Each user has an account associated with the blockchain, which allows them to pay mutual compensations to each other. The encrypted energy bill data and group purchasing decisions are also recorded on the blockchain for later validation.

The blockchain consists of several components:

(1) **Ledger**: An append-only ledger on a blockchain holds the records of all accounts and transactions. Note that by default, there is no privacy protection to the ledger, such that the account details and transaction histories are visible to the public. On Ethereum, one can create tokens (ERC20 [21]) on the ledger to represent certain digital assets. Our mutual compensation payment system is implemented by ERC20 tokens, which allows us to incorporate privacy protection. To make payment among each other, users are required to purchase tokens that will be subsequently transferred to each other and redeemed.
(2) **Accounts**: An account is identified by a public key K^{p} and an address $\mathtt{ad}$, which is the hash of the public key: $\mathtt{ad} = \mathcal{H}(K^{\mathsf{p}})$, where $\mathcal{H}(\cdot)$ is a cryptographic hash function. The user manages the account by a private key K^{s}. Each account holds a balance of tokens, denoted by $\mathtt{Bal}(\mathtt{ad})$, which by

default is a publicly visible plaintext. Each user i has an account associated with a tuple $(\mathtt{ad}_i, K_i^{\mathsf{p}}, K_i^{\mathsf{s}}, \mathtt{Bal}(\mathtt{ad}_i))$.

(3) **Transactions**: To initiate a transaction of tokens from $\mathtt{ad}_i$ to $\mathtt{ad}_{i'}$ with transaction value $\mathtt{val}$, a user submits a transaction request to the blockchain: $\mathtt{tx} = (\mathtt{ad}_i, \mathtt{ad}_{i'}, \mathtt{val})$, along with a signature $\mathtt{sign}_{K_i^{\mathsf{s}}}(\mathtt{tx})$ using the private key K_i^{s} associated with $\mathtt{ad}_i$. The transaction request will be executed[1] if $\mathtt{Bal}(\mathtt{ad}_i) \geq \mathtt{val}$.

(4) **Data**: A small amount of data can also be recorded by smart contracts on the blockchain ledger, which is visible to the public. We will introduce a privacy-preserving compact approach of data representation by cryptographic commitments and Merkel trees in the next section.

4.2 Energy Bill Data

4.2.1 *Cryptographic Commitments.*

A (cryptographic) commitment allows a user to hide a secret (e.g., to hide the balances and transactions on a blockchain), as known as Pedersen commitment. Denote a finite field by $\mathbb{Z}_p$. To commit secret value $x \in \mathbb{Z}_p$, a user first picks a random number $\mathtt{r} \in \mathbb{Z}_p$ to mask the commitment, and then computes the commitment by:

$$\mathtt{Cm}(x, \mathtt{r}) = g^x \cdot h^{\mathtt{r}} \pmod p \tag{14}$$

where g, h are generator of a multiplicative group $\mathbb{Z}_p^*$, and p is a large prime number. Pedersen commitment is perfectly hiding (i.e., no adversary can unlock the secret) and computationally binding (i.e., no adversary can associate with another secret in polynomial time). Note that Pedersen commitment satisfies homomorphic property: $\mathtt{Cm}(x_1 + x_2, \mathtt{r}_1 + \mathtt{r}_2) = \mathtt{Cm}(x_1, \mathtt{r}_1) \cdot \mathtt{Cm}(x_2, \mathtt{r}_2)$. Sometimes, we simply write $\mathtt{Cm}(x)$ without specifying random $\mathtt{r}$.

4.2.2 *Merkel Trees.*

To reduce the ledger storage space, we use a Merkel tree for data representation and only a small-sized hash pointer (i.e., the root of the Merkel tree) will be recorded on the ledger. A Merkle tree is a binary tree of hash pointers, such that the data value of a non-leave node is $\mathtt{Hash}_{\mathtt{XY}} \triangleq \mathcal{H}(\mathtt{X} \mid \mathtt{Y})$, where $\mathtt{X}, \mathtt{Y}$ are the data values of its left and right children respectively (which can also be hash pointers). A hash pointer is a concise way of data representation. The root of Merkle tree, denoted by $\mathtt{Root}$ represents the hash pointer of the whole tree. We associate each leaf in the Merkle tree by a commitment. Note that only a concise proof is needed to prove that a given data value of a leaf is a member in a given Merkle tree. For example, to prove the membership of a leaf, $\mathtt{Cm}(a)$, in the Merkle tree in Figure 3 with $\mathtt{Root}$, one only needs to locally record $\mathtt{Hash}_1, \mathtt{Hash}_{23}$ as a proof, highlighted in bold orange nodes in Figure 3, with which one can respectively reconstruct the hashes $(\mathtt{Hash}_{\emptyset}, \mathtt{Hash}_{\emptyset 1})$, up to $\mathtt{Root}$. Merkle tree eliminates the need of storing the entire dataset, but only the root. In this case, the prover only needs to record an $O(\log n)$-sized proof for each data value.

4.2.3 *Encrypted Energy Bill Receipts.*

[1]A blockchain transaction has more security elements, like nonce to prevent replay attack, account-locking against front-running attack, etc. But our model can easily incorporate these elements in practice.

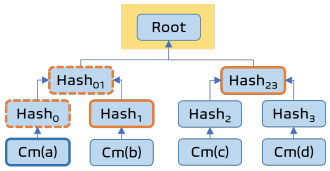

Figure 3: An example of a given Merkle tree and a proof of membership for $\mathtt{Cm}(a)$ in the Merkle tree is $(\mathtt{Hash}_1, \mathtt{Hash}_{23})$, such that $\mathtt{Root} = \mathcal{H}(\mathcal{H}(\mathcal{H}(\mathtt{Cm}(a)) \mid \mathtt{Hash}_1) \mid \mathtt{Hash}_{23})$

We next use commitments and Merkle trees to record the encrypted energy bill receipts. Recall that each user i has energy demands a_i^t at timeslot t. Let an indicator variable be $\beta_i^t \triangleq \mathbb{1}\{a_i^t \geq 0\}$, and the corresponding operational energy cost at timeslot t be $\kappa_i^t \triangleq \mathtt{Cost}_{\mathsf{Op}}^t[x_i^t, a_i^t]$. Let the potential connection and disconnection fees of each user i's current energy plan at timeslot t be

$$\mu_i^t \triangleq \mathtt{c}(x_i^t), \qquad \nu_i^t \triangleq \mathtt{d}(x_i^{t-1}) \cdot \mathbb{1}\{\mathtt{T}_i^t \leq \mathtt{T}(x_i^{t-1})\}$$

After the user paying the energy bill, the utility operator generates an encrypted receipt as follows: For a fixed epoch $[t_0, t_1]$, the utility operator records the following tuple for each timeslot $t \in [t_0, t_1]$ of the user:

$$\left(\mathtt{Cm}(a_i^t), \mathtt{Cm}(\beta_i^t), \mathtt{Cm}(\kappa_i^t), \mathtt{Cm}(\mu_i^t), \mathtt{Cm}(\nu_i^t)\right)_{t=t_0}^{t_1}$$

These tuples will be represented by leaves in a Merkle tree, and its root with a verifiable signature and a timestamp will be recorded on the ledger. In the next sequent, we will utilize multi-party computation for validating if the shared energy bills data matches the encrypted receipts on the ledger.

5 MULTI-PARTY COMPUTATION PROTOCOL

We presented a decision mechanism for group purchasing in Section 2.5. In this section, we describe the way to execute the mechanism in a decentralized privacy-preserving manner by multi-party computation. This section presents a simplified multi-party computation protocol called SPDZ [17, 18], which allows multiple parties to jointly compute a certain function with concealed input. SPDZ can safeguard against a dishonest majority (i.e., all but one party may be dishonest), and does not require a trusted setup.

The full details of SPDZ protocol can be found in Appendix. C. Here, we only sketch some high-level ideas of SPDZ, and show how to use SPDZ to validate secretly shared private input by cryptographic commitments on the blockchain ledger.

5.1 Basic Operations of SPDZ

In a nutshell, SPDZ relies on secret-sharing, whereby private data will be distributed to multiple parties, such that each party only knows a share of the data, without complete knowledge of other shares. Thus, the computation of individual shares of data will not reveal the original data, unless all shares are revealed for output or validation. Several distributed computation operations can be performed locally, while preserving the secret-sharing property.

Suppose a private number x is distributed to n parties, such that each party i knows a share x_i only and $x = \sum_{i=1}^n x_i$, but not knowing other shares x_j for $j \neq i$. Note that no one is unable to construct x, without knowing all the shares. In the following, we

write $\langle x \rangle$ as a *secretly shared* number, meaning that there is a vector $(x_1, ..., x_n)$, such that each party i knows only x_i. Given secretly shared $\langle x \rangle$ and $\langle y \rangle$, and a public known constant c, we can compute the following operations in a privacy-preserving manner by local computation at each party, and then the outcome can be assembled from the individual shares:

A1) $\langle x \rangle + \langle y \rangle$ can be computed locally by $(x_1 + y_1, ..., x_n + y_n)$.
A2) $c \cdot \langle x \rangle$ can be computed locally by $(c \cdot x_1, ..., c \cdot x_n)$.
A3) $c + \langle x \rangle$ can be computed locally by $(c + x_1, x_2, ..., x_n)$.

To reveal $\langle x \rangle$, each party i simply broadcasts x_i to other parties. Then each party can reconstruct $x = \sum_{i=1}^{n} x_i$. Note that multiplications can also be computed in a privacy-preserving manner, albeit with a more complex setup (see Appendix. C). With additions and multiplications, one can compute a large class of functions (including comparison and branching conditions).

Note that some parties may be dishonest, who may not perform the required local computations correctly. We need to safeguard against dishonest parties by validation through message authentication codes (MACs). Every secretly shared number is encoded by a MAC as $\gamma(x)$, which is also secretly shared as $\langle \gamma(x) \rangle$. The basic idea is that if a dishonest party wants to modify his share x_i, then he also needs to modify $\gamma(x)_i$ consistently. This allows dishonesty to be detectable by checking the corresponding MAC in the final output. In the following, we write $\langle\langle x \rangle\rangle$ meaning that both $\langle x \rangle$ and the respective MAC $\langle \gamma(x) \rangle$ are secretly shared among the users.

We next sketch a high-level description of SPDZ protocol:

(1) *Pre-processing Phase*: In this phase, a collection of shared random numbers will be constructed that can be used to mask the private input numbers. For each private input number of party i, there needs a shared random number $\langle\langle r^i \rangle\rangle$, where r^i is revealed to party i only, but not to other parties.
(2) *Online Phase*: To secretly shares a private input number x^i using $\langle\langle r^i \rangle\rangle$, without revealing x^i, it proceeds as follows:
   1) Party i computes and reveals $z^i = x^i - r^i$ to all parties.
   2) Every party sets $\langle\langle x^i \rangle\rangle \leftarrow z^i + \langle\langle r^i \rangle\rangle$ (see A3).

   Next, any computation functions in terms of additions or multiplications can be computed by proper local computations (e.g., A1-A3). The MACs are updated accordingly to preserve the consistency. See Appendix. C for details.
(3) *Output and Validation Phase*: All MACs will be revealed for validation. If there is any inconsistency in MACs, then abort.

5.2 Decentralized Validation of Energy Bill Data

With SPDZ, one can perform the decision-making mechanism of Section 2.5 in a decentralized privacy-preserving manner. We will provide the detailed procedures in SPDZ in the next section. In the following, we particularly explain the joint validation of encrypted private input data on the blockchain ledger via SPDZ.

On one hand, each user i secretly shares her energy bill data $\big(\langle\langle a_i^t \rangle\rangle, \langle\langle \beta_i^t \rangle\rangle, \langle\langle \kappa_i^t \rangle\rangle, \langle\langle \mu_i^t \rangle\rangle, \langle\langle \nu_i^t \rangle\rangle\big)$ as the input to the decision-making mechanism. On the other hand, user i proves that the corresponding $\big(\mathtt{Cm}(a_i^t), \mathtt{Cm}(\beta_i^t), \mathtt{Cm}(\kappa_i^t), \mathtt{Cm}(\mu_i^t), \mathtt{Cm}(\nu_i^t)\big)$ are recorded by a root of a Merkle tree on the ledger with appropriate signature and timestamp from the utility operator. The proof of a membership in a Merkle tree is attained by providing a valid path in the Merkle tree that matches the root.

However, there may be inconsistency between these inputs of a dishonest user, namely, the secretly shared values may not match the commitments. Hence, to ensure the consistency, the users generate ZKPs to prove the knowledge in the commitments from the secretly shared values. For example, user i generates a ZKP for the proof of knowledge of a_i^t in $\mathtt{Cm}(a_i^t)$ from $\langle\langle a_i^t \rangle\rangle$, without disclosing a_i^t. Note that there is a well-known technique (called Σ-protocol; see Appendix B) to generate a ZKP from a private value x for a given $\mathtt{Cm}(x, \mathsf{r})$. We briefly review the its construction as follows:

(1) The prover generates a pair of random numbers (x', r') and announces its commitment $\mathtt{Cm}(x', \mathsf{r}')$ to the verifier.
(2) The verifier generates a random challenge ψ and announces ψ to the prover.
(3) The prover computes $z_x \leftarrow x' + \psi \cdot x$ and $z_\mathsf{r} \leftarrow \mathsf{r}' + \psi \cdot \mathsf{r}$, and announces (z_x, z_r) to the verifier.
(4) The verifier checks the condition: If $g^{z_x} \cdot h^{z_\mathsf{r}} = \mathtt{Cm}(x', \mathsf{r}') \cdot \mathtt{Cm}(x, \mathsf{r})^{\psi}$ (which is based on the homomorphic property of Pedersen commitments), then it passes the verification.

Next, we replace the above construction of ZKP by a distributed version via SPDZ, denoted by $\Pi_{\mathtt{dzkpCm}}$ (see Algorithm 1). Since $\Pi_{\mathtt{dzkpCm}}$ is carried out jointly by all users without disclosing the secretly shared input, passing the verification will justify the consistency between the secretly shared values and the commitments. In $\Pi_{\mathtt{dzkpCm}}$, the random challenge ψ is generated by random strings from all users. Also, the MACs in SPDZ will ensure that the distributed computations of $\Pi_{\mathtt{dzkpCm}}$ are performed correctly, despite the presence of dishonest users.

Algorithm 1 $\Pi_{\mathtt{dzkpCm}}$: *Prove the knowledge of x in a given commitment $\mathtt{Cm}(x)$ from secretly shared $\langle\langle x \rangle\rangle$ via SPDZ*

Input: $\mathtt{Cm}(x, \mathsf{r})$ (known to all users), $\langle\langle x \rangle\rangle$ (already secretly shared among users); (x, r) (known to user i only)
Output: Pass or Fail
1: User i announces $\mathtt{Cm}(x, \mathsf{r})$ and secretly shares $\langle\langle \mathsf{r} \rangle\rangle$ with all users
2: User i generates a pair of random numbers (x', r') and announces $\mathtt{Cm}(x', \mathsf{r}')$ to all users
3: User i secretly shares $\langle\langle x' \rangle\rangle$ and $\langle\langle \mathsf{r}' \rangle\rangle$ with all users
4: All users conduct a coin-tossing protocol to obtain a random challenge ψ as follows:
   (1) Each user j announces a commitment $\mathtt{Cm}(\mathsf{r}_j'')$ of a random string r_j''
   (2) After receiving the commitments $\{\mathtt{Cm}(\mathsf{r}_j'')\}$ from all users, each user reveals r_j'' to all users and all users check if it matches $\mathtt{Cm}(\mathsf{r}_j'')$
   (3) Set $\psi \leftarrow \mathcal{H}(\mathsf{r}_1''|...|\mathsf{r}_{|X|}'')$ ▷ *Generate random challenge by hashing concatenated random strings*
5: Compute the following via SPDZ among all users:

$$\langle\langle z_x \rangle\rangle \leftarrow \langle\langle x' \rangle\rangle + \psi \cdot \langle\langle x \rangle\rangle$$
$$\langle\langle z_\mathsf{r} \rangle\rangle \leftarrow \langle\langle \mathsf{r}' \rangle\rangle + \psi \cdot \langle\langle \mathsf{r} \rangle\rangle$$

6: Reveal z_x, z_r and their MACs to all users

▷ *All users check the following*
7: **if** $g^{z_x} \cdot h^{z_\mathsf{r}} = \mathtt{Cm}(x', \mathsf{r}') \cdot \mathtt{Cm}(x, \mathsf{r})^{\psi}$ and checking all revealed MACs passed **then**
8: **return** Pass
9: **else**
10: **return** Fail and abort
11: **end if**

▷ *The distributed ZKP of commitment is* $\mathtt{zkpCm}[x, \mathtt{Cm}(x)] = \big\{\mathtt{Cm}(x, \mathsf{r}); \mathtt{Cm}(x', \mathsf{r}'), (\mathsf{r}_i'')_{i \in X}, z_x, z_\mathsf{r}\big\}$

6 PRIVACY-PRESERVING MECHANISM AND PROTOCOLS

Based on the results of the previous sections, this section presents the full decision-making mechanism for group purchasing with privacy-preserving protocols based on SPDZ and ZKPs. For the

brevity of presentation, we use the following shorthand notations in Table 1 to represent various energy costs and fees.

Meaning	Notation	Representation
Consumption indicator	β_i^t	$= \mathbb{1}\{a_i^t \geq 0\}$
Operational energy cost of plan x_i^t	κ_i^t	$= \mathtt{Cost}_{\mathsf{Op}}^t[x_i^t, a_i^t]$
Connection fee of plan x_i^t	μ_i^t	$= \mathtt{c}(x_i^t)$
Disconnection fee of plan x_i^t	ν_i^t	$= \mathtt{d}(x_i^{t-1}) \cdot \mathbb{1}\{\mathsf{T}_i^t \leq \mathsf{T}(x_i^{t-1})\}$
Operational energy cost of plan $\mathfrak{g}$	$\kappa_{i,\mathfrak{g}}^t$	$= \mathtt{Cost}_{\mathsf{Op}}^t[\mathfrak{g}, a_i^t]$
Offline optimal cost of plan x_i^t at t	$\mathtt{Opt}_i^t$	$= \mathtt{Opt}_i^t[x_i^t]$
Offline optimal cost of plan $\mathfrak{g}$ at t	$\mathtt{Opt}_{i,\mathfrak{g}}^t$	$= \mathtt{Opt}_i^t[\mathfrak{g}]$

Table 1: Table of shorthand notations.

To simplify the presentation, we only consider the switching decision for a group of users X from their individual energy plans to a group-based plan that is decided by WFA on the social problem ($\mathsf{P}_{\mathrm{soc}}$). We first introduce some protocols as sub-routines for several common distributed computation tasks via SPDZ:

(1) $\Pi_{\min}[\langle\langle y\rangle\rangle, \langle\langle z\rangle\rangle]$ is a protocol that computes secretly shared output $\langle\langle x\rangle\rangle$ via SPDZ, given two secretly shared values $\langle\langle y\rangle\rangle, \langle\langle z\rangle\rangle$, such that $\langle\langle x\rangle\rangle \leftarrow \min\{\langle\langle y\rangle\rangle, \langle\langle z\rangle\rangle\}$.
(2) $\Pi_{<}[\langle\langle x\rangle\rangle, \langle\langle y\rangle\rangle]$ is a protocol that compares two secretly shared values $\langle\langle x\rangle\rangle$ and $\langle\langle y\rangle\rangle$ via SPDZ, and outputs 1 if $\langle\langle x\rangle\rangle - \langle\langle y\rangle\rangle < 0$, and 0 otherwise, without revealing x, y.
(3) $\Pi_{=}[\langle\langle x\rangle\rangle, \langle\langle y\rangle\rangle]$ is a protocol that compares two secretly shared values $\langle\langle x\rangle\rangle$ and $\langle\langle y\rangle\rangle$via SPDZ, and outputs 1 if $\langle\langle x\rangle\rangle - \langle\langle y\rangle\rangle = 0$, and 0 otherwise, without revealing x, y.

The details of these protocols can be found in Appendix D.1.

We divide the decision-making mechanism for group purchasing into four main stages as follows:

Stage 0: (Registration and Initialization)

First, a group-based energy plan $\mathfrak{g}$ is registered on the blockchain. There are a group of users X (such that $|X| \geq \mathsf{N}(\mathfrak{g})$), who are interested in plan $\mathfrak{g}$. To begin the decision-making process for group purchasing, there are various initialization processes of the protocols, for instance, the pre-processing phase of SPDZ (see Appendix. C). After the initialization, the users will proceed to Stage 1.

Stage 1: (Data Sharing and Validation)

We suppose that the energy bills are charged every fixed epoch. At each epoch, the users provide the energy bill data of the current epoch for the group purchasing decision-making process in a privacy-preserving manner. To generate the private input, the users secretly share their energy bill data of the current epoch $\big(\langle\langle a_i^t\rangle\rangle, \langle\langle \beta_i^t\rangle\rangle, \langle\langle \kappa_i^t\rangle\rangle, \langle\langle \mu_i^t\rangle\rangle, \langle\langle \nu_i^t\rangle\rangle\big)$ for the epoch $t \in [t_0, t_1]$ with others by following the procedure of the online phase in SPDZ. The users also need to validate that their secretly-shared energy bill data matches the encrypted commitments $\big(\mathtt{Cm}(a_i^t), \mathtt{Cm}(\beta_i^t), \mathtt{Cm}(\kappa_i^t), \mathtt{Cm}(\mu_i^t), \mathtt{Cm}(\nu_i^t)\big)$ on the blockchain ledger by $\Pi_{\mathtt{dzkpCm}}$, as described in Section 5.2. Note that these commitments are prepared by the utility operators. Hence, we assume the validity of the committed values. If all secretly-shared energy bill data passes the validation of $\Pi_{\mathtt{dzkpCm}}$, the users will proceed to Stage 2.

Stage 2: (Decision-Making)

In this stage, the users jointly compute the online decision-making algorithm (i.e., WFA on the social problem ($\mathsf{P}_{\mathrm{soc}}$)) in a privacy-preserving manner via SPDZ. It is possible that the users continue the online decision-making algorithm from the last epoch, if the last epoch's decision is not to join the group-based plan $\mathfrak{g}$. In this case, the offline optimal costs of the last epoch, $\langle\langle \mathtt{Opt}_i^t\rangle\rangle$, will also be input to the online decision-making algorithm.

The users execute WFA by a distributed protocol via SPDZ. The protocol is described by Π_{WFA} (Algorithm 2), in which the users first compute the offline optimal costs $\langle\langle \mathtt{Opt}_i^t\rangle\rangle$ and $\langle\langle \mathtt{Opt}_{i,\mathfrak{g}}^t\rangle\rangle$ in dynamic programming via SPDZ. Then the users will test conditions in WFA via SPDZ whether they should join energy plan $\mathfrak{g}$ or not. Π_{WFA} calls sub-routines $\Pi_{\min}, \Pi_{<}, \Pi_{=}$ to complete the tasks.

If the decision is not to join plan $\mathfrak{g}$ at the current epoch, then the users will return to Stage 1 for the next epoch. Otherwise, the users will proceed to Stage 3, and compute the mutual compensations and carry out the payments on blockchain.

Algorithm 2 Π_{WFA}: *Compute the group purchasing decision by WFA via SPDZ whether the users in X should join energy plan $\mathfrak{g}$*

Input: $\Big(\big(\langle\langle a_i^t\rangle\rangle, \langle\langle \beta_i^t\rangle\rangle, \langle\langle \kappa_i^t\rangle\rangle, \langle\langle \mu_i^t\rangle\rangle, \langle\langle \nu_i^t\rangle\rangle\big)_{t=t_0}^{t_1}\Big)_{i\in X}, \big(\langle\langle \mathtt{Opt}_i^{t_0-1}\rangle\rangle\big)_{i\in X}$
Output: Join or Stay
for $t \in [t_0, t_1]$ **do**
  for $i \in X$ **do**
    Compute the following via SPDZ among all users:
$$\langle\langle \kappa_{i,\mathfrak{g}}^t\rangle\rangle \leftarrow \mathtt{p}_+^t(\mathfrak{g}) \cdot \langle\langle a_i^t\rangle\rangle \cdot \langle\langle \beta_i^t\rangle\rangle + \mathtt{p}_-^t(\mathfrak{g}) \cdot \langle\langle a_i^t\rangle\rangle \cdot (1 - \langle\langle \beta_i^t\rangle\rangle)$$
    Compute the following offline optimal costs via SPDZ by calling protocol $\Pi_{\min}$:
$$\langle\langle \mathtt{Opt}_i^t\rangle\rangle \leftarrow \Pi_{\min}\Big[\langle\langle \mathtt{Opt}_i^{t-1}\rangle\rangle + \langle\langle \kappa_i^t\rangle\rangle,\ \langle\langle \mathtt{Opt}_{i,\mathfrak{g}}^{t-1}\rangle\rangle + \langle\langle \kappa_i^t\rangle\rangle + \langle\langle \mu_i^t\rangle\rangle + \mathtt{d}(\mathfrak{g})\Big]$$
$$\langle\langle \mathtt{Opt}_{i,\mathfrak{g}}^t\rangle\rangle \leftarrow \Pi_{\min}\Big[\langle\langle \mathtt{Opt}_{i,\mathfrak{g}}^{t-1}\rangle\rangle + \langle\langle \kappa_{i,\mathfrak{g}}^t\rangle\rangle,\ \langle\langle \mathtt{Opt}_i^{t-1}\rangle\rangle + \langle\langle \kappa_{i,\mathfrak{g}}^t\rangle\rangle + \mathtt{c}(\mathfrak{g}) + \langle\langle \nu_i^t\rangle\rangle\Big]$$
  end for
  ▷ *Apply WFA to decide if switching to group plan $\mathfrak{g}$. Call $\Pi_{<}$ and $\Pi_{=}$ to test the conditions*
  if $\Pi_{<}\Big[\sum_{i\in X}\langle\langle \mathtt{Opt}_{i,\mathfrak{g}}^t\rangle\rangle + \mathtt{c}(\mathfrak{g}) + \langle\langle \nu_i^t\rangle\rangle, \sum_{i\in X}\langle\langle \mathtt{Opt}_i^t\rangle\rangle\Big] = 1$ and
    $\Pi_{=}\Big[\sum_{i\in X}\langle\langle \mathtt{Opt}_{i,\mathfrak{g}}^t\rangle\rangle, \sum_{i\in X}\langle\langle \mathtt{Opt}_{i,\mathfrak{g}}^{t-1}\rangle\rangle + \langle\langle \kappa_{i,\mathfrak{g}}^t\rangle\rangle\Big] = 1$ **then**
    return Join
  end if
end for
return Stay

Stage 3: (Mutual Compensations)

After deciding to join plan $\mathfrak{g}$, the users compute their mutual compensations in a privacy-preserving manner via SPDZ based on the secretly-shared input from Stage 2. They will pay the mutual compensations on the blockchain ledger afterwards and the decision will be recorded on the ledger for the energy provider of plan $\mathfrak{g}$.

The users compute mutual compensations by a distributed protocol via SPDZ. The protocol is described by Π_{MC} (Algorithm 3), in which the users compute the compensated switching cost $\big(\langle\langle \theta_i\rangle\rangle\big)_{i\in X}$ by Theorem 3, and the net mutual compensations (ϕ_i) between the compensated and original switching costs, defined by $\phi_i \triangleq \theta_i - \mathtt{Cost}_{\mathsf{Sw}}^t[x_i^{t-1}, \mathfrak{g}]$. In this stage, $\big(\langle\langle \phi_i\rangle\rangle\big)_{i\in X}$ will remain secretly-shared, and will be used in Stage 4 for payments on blockchain.

Stage 4: (Payments)

The users pay the net mutual compensations $(\phi_i)_{i\in X}$ by ERC20 tokens on the blockchain. The users generate a multi-transaction request with concealed transaction values $\big(\mathtt{Cm}(\phi_i)\big)_{i\in X}$. The transaction generation will be handled in a privacy-preserving manner via SPDZ, which follows a similar privacy-preserving payment system in [37]. Because of the paucity of space, we skip the protocol, and defer the details to Appendix D.2.

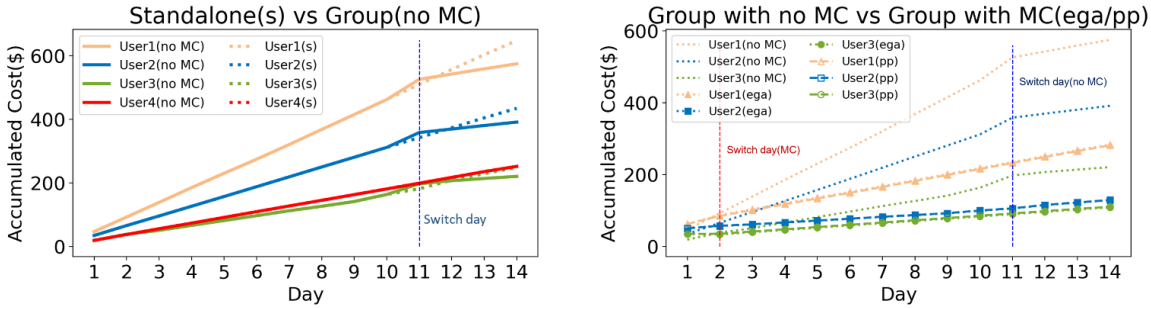

Figure 4: Evaluation of online decision algorithm, comparing the online solutions with and without mutual compensations.

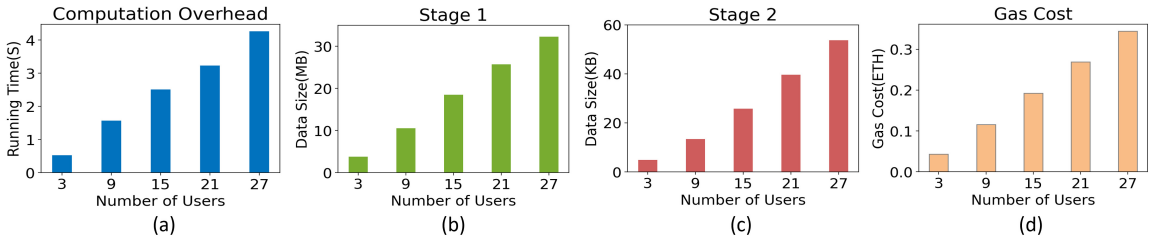

Figure 5: SPDZ performance and smart contract gas costs

Algorithm 3 Π_{MC}: *Compute compensated switching cost* $\left(\langle\!\langle \theta_i \rangle\!\rangle\right)_{i\in X}$ *according to either egalitarian cost-sharing or proportional cost-sharing and return the net mutual compensations*

Input: $\left(\langle\!\langle v_i^t \rangle\!\rangle, \langle\!\langle \texttt{Opt}_i^t \rangle\!\rangle, \langle\!\langle \texttt{Opt}_{i,\mathfrak{g}}^t \rangle\!\rangle\right)_{i\in X}$
Output: $\left(\langle\!\langle \theta_i \rangle\!\rangle\right)_{i\in X}$
for $i \in X$ **do**
  Compute the following via SPDZ among all users:

$$\langle\!\langle C_i^t(\mathcal{E}\backslash\{\mathfrak{g}\}) \rangle\!\rangle \leftarrow \langle\!\langle \texttt{Opt}_i^t \rangle\!\rangle \tag{15}$$

$$\langle\!\langle C_i^t(\mathfrak{g}) \leftarrow \langle\!\langle \texttt{Opt}_{i,\mathfrak{g}}^t \rangle\!\rangle + \mathsf{c}(\mathfrak{g}) + \langle\!\langle v_i^t \rangle\!\rangle \tag{16}$$

end for
Compute and reveal $\sum_{i\in X}\langle\!\langle C_i^t(\mathcal{E}\backslash\{\mathfrak{g}\}) \rangle\!\rangle$, $\sum_{i\in X}\langle\!\langle C_i^t(\mathfrak{g}) \rangle\!\rangle$, and their MACs to all users.
if Checking MACs failed **then**
  Abort
end if
for $i \in X$ **do**
  Compute the following via SPDZ among all users:

$$\langle\!\langle \theta_i \rangle\!\rangle \leftarrow \begin{cases} \langle\!\langle C_i^t(\mathcal{E}\backslash\{\mathfrak{g}\}) \rangle\!\rangle + \frac{\sum_{i'\in X} C_{i'}^t(\mathfrak{g}) - \sum_{i'\in X} C_{i'}^t(\mathcal{E}\backslash\{\mathfrak{g}\})}{|X|} - \langle\!\langle \texttt{Opt}_{i,\mathfrak{g}}^t \rangle\!\rangle, & \text{for ega. cost-sharing} \\ \langle\!\langle C_i^t(\mathcal{E}\backslash\{\mathfrak{g}\}) \rangle\!\rangle \cdot \frac{\sum_{i'\in X} C_{i'}^t(\mathfrak{g})}{\sum_{i'\in X} C_{i'}^t(\mathcal{E}\backslash\{\mathfrak{g}\})} - \langle\!\langle \texttt{Opt}_i^t[\mathfrak{g}] \rangle\!\rangle, & \text{for prop. cost-sharing} \end{cases}$$

end for
▷ *Compute the net mutual compensations via SPDZ*
for $i \in X$ **do**
  $\langle\!\langle \phi_i \rangle\!\rangle \leftarrow \langle\!\langle \theta_i \rangle\!\rangle - \mathsf{c}(\mathfrak{g}) - \langle\!\langle v_i^t \rangle\!\rangle$
  Reveal $\langle\!\langle \phi_i \rangle\!\rangle$ and its MAC to user i only
  if Checking MAC failed **then**
    Abort
  end if
end for

7 EVALUATION

This section presents an empirical evaluation of our solution, including the online decision-making algorithm, SPDZ performance and the gas costs of smart contract on Ethereum.

7.1 Online Decision-Making Algorithm

We present an evaluation of our online algorithm using real-world hourly electricity consumption data of 4 users (see Figure 6 and User 3 has a solar PV that can probably export extra electricity to the grid). The data trace represents the total daily electricity usage of each household in a period of consecutive 14 days. The energy plan tariffs in table 2 are characterized by peak hour rates, off-peak hour rates and disconnection fees. The peak hour period is from 8am to 8pm, and the off-peak hour period is from 9pm to 7am respectively. We set a zero connection fee in our experiments. We evaluated the performance of the online decision algorithm without mutual compensations. In Figure 4, we observe a small spike in the accumulated cost on Day 11 among Users 1, 2 and 3 due to switching to the group plan and paying off the disconnection fee of the standard plan. But the users enjoy more cost-saving afterwards. We also study different mutual compensation schemes. We observe that User 1 pays more with egalitarian cost-sharing (ega) to compensate user 2 and user 3, whereas Users 1 and 2 pay more with proportional cost-sharing (pp) to compensate user 3 in the beginning, but all of them will be benefited considerably in a long term. At last, we study the performance of our online algorithm considering with and without mutual compensations (MC). We observe that with mutual compensations (MC), all users join the group-based plan earlier on Day 2. Hence, the energy costs of all users are lower than those without mutual compensations. All users are benefited substantially with more than 50% cost reduction, when joining the group-based plan earlier with mutual compensations.

Table 2: Table of Energy Plan Tariffs.

	Peak ($)	Off-Peak ($)	Disconnect Fee ($)
Standalone Plan	1.6	1.0	16
Group Plan	0.6	0.3	30

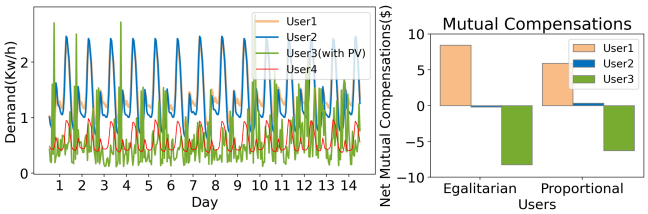

Figure 6: Data trace of energy consumption and net mutual compensations.

7.2 SPDZ Performance

We evaluated the performance of SPDZ in Stage 2. We do not show the performance of Stage 3 and Stage 4, because of their small overhead. We consider a single fortnight period and up to 27 users.

Computation Running Time: All the results are averaged over 10 instances. Figure 5 represents the average running time of each user in Stage 2. We observe that the running time only increases linearly with the number of users with moderate overhead.

Communication Overhead: We evaluated the communication data size among Stage 1 and Stage 2. Figure 5 shows that the total communication data size grows linearly with the number of users in both Stage 1 and Stage 2 with moderate overhead.

7.3 Smart Contract Gas Costs

We implemented the payment system of mutual compensations by a smart contract on Ethereum using Solidity programming language. We measured the gas costs of our smart contract, it represents the amount of computational resources needed to execute a single transaction. The gas price is set by the operator, and most miners will choose the smart contract with a higher gas price in a block. We use the standard gas price 54 Gwei to simulate the transaction cost in Ether. Figure 5 shows the gas costs. We observe that the gas cost increases linearly, starting from 0.0437 ETH with 3 users to 0.3445 ETH with 27 users. Overall, our smart contract generated comparable gas costs as other privacy-preserving smart contract implementations in the literature.

8 RELATED WORK

In this section, we present the related work in the literature.

The problem of energy plan selection has been studied as an online decision problem, which belongs to the class of Metrical Task System (MTS) problem [7] and online convex optimization (OCO) problem with switching cost [4]. For a general setting with n discrete states, the MTS problem is known to have a competitive ratio of $2n - 1$ [2]. An instance of MTS is the energy generation scheduling in microgrids [25, 29, 31]. [1, 28] proposed online algorithms for OCO. The problem of energy plan selection is also studied as an MTS problem in [38]. Most of the extant literature of online decision problems does not consider the settings of cooperative multi-players, in which the players may decide their joint online decisions. Recently, the work in [41] considers cooperative multi-player online decision problems of group purchasing energy plan selection problem. This paper extends the online algorithm with mutual compensations in [41] by considering a different mutual compensation scheme and the possibility of energy export.

Blockchain technology has been applied to diverse aspects of energy systems. Among the applications, [24] applied blockchain to mitigate trust in peer-to-peer electric vehicle charging. Blockchain has been applied to microgrid energy exchange and wholesale markets by prosumers [32]. Renewable energy credits and emissions trading are also applications of blockchain [27]. In these applications, the goal of blockchain is to improve transparency and reduce settlement times, since blockchain can ensure integrity and consistency of transactions and settlement on an open ledger. See a recent survey about blockchain applications to energy systems [3]. However, very few studies considered the privacy of blockchain, even transaction data on the ledger is entirely disclosed to the public. [37] recently proposed a privacy-preserving solution for energy storage sharing using blockchain and secure multi-party computation. This work draws on similar concepts from privacy-preserving blockchain, but also addressing a different problem of decentralized group purchasing of energy plans. It is worth noting that supporting privacy on blockchain is a crucial research topic in cryptography and security. There are several privacy-preserving blockchain platforms with support of privacy (e.g., ZCash, Monero, Zether, Tornado Cash [6, 9, 33]).

There are two major approaches in privacy-preserving techniques in the literature: (1) data obfuscation that masks private data with random noise, (2) secure multi-party computation that hides private data while allowing the data to be computed confidentially. Differential privacy [20], a main example of data obfuscation, is often used in privacy-preserving data mining in a very large dataset [39]. Note that there is an intrinsic trade-off between accuracy and privacy in differential privacy. On the other hand, secure multi-party computation [19, 23] traditionally employed garbled circuits [26], which have a high computational complexity, and homomorphic cryptosystems [16, 30, 35], which need a trusted setup for key generation. Recently, information-theoretical secret-sharing [17, 18] has been utilized for secure multi-party computation, which provides high efficiency and requires no trusted third-party setup. Particularly, SPDZ [17] is an efficient MPC protocol based on secret-sharing and can safeguard against a dishonest majority of users.

9 CONCLUSION

The emergence of competitive retail energy markets introduces group purchasing of energy plans. However, traditional group purchasing mediated by a trusted third-party suffers from the lack of privacy and transparency. In this paper, we introduced a novel paradigm of *decentralized privacy-preserving* group purchasing. We combine competitive online algorithms, secure multi-party computations and zero-knowledge on blockchain as a holistic solution to enable users to make coordinated switch decisions in a decentralized manner, without a trusted third-party. We implemented our decentralized group purchasing solution by multi-party computation protocol (SPDZ) and a smart contract on Solidity-supported blockchain platform. Although we implemented it as a smart contract on permissionless blockchain, it can be implemented on a permissioned blockchain, on which the gas cost is not a concern. In future work, we will extend our online decision algorithms to complex scenarios (e.g., with termination penalty in group purchasing).

REFERENCES

[1] Susanne Albers and Jens Quedenfeld. 2018. Optimal algorithms for right-sizing data centers. *SPAA* (2018).

[2] Borodin Allan, Nathan Linial, and Michael E. Saks. 1992. An optimal on-line algorithm for metrical task system. *JACM* (1992).

[3] Merlinda Andoni, Valentin Robu, David Flynn, Simone Abram, Dale Geach, David P. Jenkins, Peter McCallum, and Andrew Peacock. 2019. Blockchain technology in the energy sector: A systematic review of challenges and opportunities. *Renewable and Sustainable Energy Reviews* 100 (2019), 143–174.

[4] Nikhil Bansal, Anupam Gupta, Ravishankar Krishnaswamy, Kirk Pruhs, Kevin Schewior, and Cliff Stein. 2015. A 2-competitive algorithm for online convex optimization with switching costs. *APPROX* (2015).

[5] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. 1988. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation. In *Annual ACM Symposium on Theory of Computing (STOC)*.

[6] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, and Eran Tromerand Madars Virza. 2014. Zerocash: Decentralized Anonymous Payments from Bitcoin. In *IEEE Symposium on Security and Privacy*.

[7] Allan Borodin and Ran El-Yaniv. 2005. *Online Computation and Competitive Analysis*. Cambridge University Press.

[8] William J. Buchanan. 2017. *Cryptography*. River Publishers.

[9] Benedikt Bunz, Shashank Agrawal, Mahdi Zamani, and Dan Boneh. 2020. Zether: Towards Privacy in a Smart Contract World. In *Financial Cryptography and Data Security (FC)*.

[10] Chi-Kin Chau and Khaled Elbassioni. 2018. Quantifying Inefficiency of Fair Cost-Sharing Mechanisms for Sharing Economy. *IEEE Trans. Control of Network System* 5 (Dec 2018), 1809–1818. Issue 4. https://arxiv.org/abs/1511.05270.

[11] Sid Chi-Kin Chau, Khaled Elbassioni, and Yue Zhou. 2022. *Approximately Socially-Optimal Decentralized Coalition Formation with Application to P2P Energy Sharing*. Technical Report. https://arxiv.org/abs/2009.08632.

[12] Sid Chi-Kin Chau, Jiajia Xu, Wilson Bow, and Khaled Elbassioni. 2019. Peer-to-Peer Energy Sharing: Effective Cost-Sharing Mechanisms and Social Efficiency. In *ACM Intl. Conf. on Future Energy Systems (e-Energy)*.

[13] Rachel R. Chen and Paolo Roma. 2011. Group buying of competing retailers. *Production and Operations Management* 20, 2 (2011), 181–197.

[14] Valerie Chen, Valerio Pastro, and Mariana Raykova. 2018. Secure Computation for Machine Learning With SPDZ. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*.

[15] ChoiceEnergy. 2021. Cut energy costs with the bulk-buying power of a group tender. https://www.choiceenergy.com.au/group-energy-tenders. (2021).

[16] Ronald Cramer, Ivan Damgård, and Jesper B Nielsen. 2001. Multiparty computation from threshold homomorphic encryption. In *Intl. conference on the theory and applications of cryptographic techniques*.

[17] Ronald Cramer, Ivan Bjerre Damgård, and Jesper Buus Nielsen. 2015. *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press. Cambridge Books Online.

[18] Ivan Damgård, Marcel Keller, Enrique Larraia, Valerio Pastro, Peter Scholl, and Nigel P. Smart. 2013. Practical Covertly Secure MPC for Dishonest Majority - or: Breaking the SPDZ Limits. In *European Symposium on Research in Computer Security (ESORICS)*.

[19] Wenliang Du and Mikhail J Atallah. 2001. Secure multi-party computation problems and their applications: a review and open problems. In *the Workshop on New Security Paradigms*.

[20] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*. Springer.

[21] Ethereum.org. 2022. ERC-20 Token Standar. https://ethereum.org/en/developers/docs/standards/tokens/erc-20/. (2022).

[22] East Coast Power & Gas. 2021. General Temrs & Conditions – New York. http://www.ecpg.com/terms. (2021).

[23] Oded Goldreich. 1998. Secure multi-party computation. *Manuscript. Preliminary version* 78 (1998).

[24] Christian Gorenflo, Lukasz Golab, and Srinivasan Keshav. 2019. Using a Blockchain to Mitigate Trust in Electric Vehicle Charging. In *ACM Intl. Conf. on Future Energy Systems (e-Energy)*.

[25] Mohammad H. Hajiesmaili, Chi-Kin Chau, Minghua Chen, and Longbu Huang. 2016. Online Microgrid Energy Generation Scheduling Revisited: The Benefits of Randomization and Interval Prediction. *ACM e-Energy* (2016).

[26] Carmit Hazay and Yehuda Lindell. 2010. *Efficient Secure Two-Party Protocols - Techniques and Constructions*. Springer.

[27] Fabian Knirsch, Clemens Brunner, Andreas Unterweger, and Dominik Engel. 2020. Decentralized and permission-less green energy certificates with GECKO. *Energy Informatics* 3, 2 (2020).

[28] Minghong Lin, Adam Wierman, Lachlan L. H. Andrew, and Eno Thereska. 2013. Dynamic right-sizing for power-proportional data centers. *IEEE/ACM TON* (2013).

[29] Lian Lu, Jinlong Tu, Chi-Kin Chau, Minghua Chen, and Xiaojun Lin. 2013. Online energy generation scheduling for microgrids with intermittent energy sources and co-generation. *ACM SIGMETRICS* (2013).

[30] Lingjuan Lyu, Sid Chi-Kin Chau, Nan Wang, and Yifeng Zheng. 2020. Cloud-based Privacy-Preserving Collaborative Consumption for Sharing Economy. *IEEE Trans. Cloud Computing* (2020).

[31] Ali Menati, Chi-Kin Chau, and Minghua Chen. 2022. Competitive Prediction-Aware Online Algorithms for Energy Generation Scheduling in Microgrids. *ACM e-Energy* (2022).

[32] Esther Mengelkamp, Johannes Garttner, Kerstin Rock, Scott Kessler, Lawrence Orsini, and Christof Weinhardt. 2018. Designing microgrid energy markets: A case study: The Brooklyn Microgrid. *Applied Energy* 210 (2018), 870–880.

[33] Monero. 2021. http://getmonero.org. (2021).

[34] The Ethereum Yellow Paper. 2014. https://ethereum.github.io/yellowpaper/paper.pdf. (2014).

[35] Eman Mohammed Radi, Noureddine Lasla, Spiridon Bakiras, and Mohamed Mahmoud. 2019. Privacy-Preserving Electric Vehicle Charging for Peer-to-Peer Energy Trading Ecosystems. In *IEEE ICC*.

[36] Australian Energy Regulator. 2021. Energy Made Easy. https://www.energymadeeasy.gov.au/. (2021).

[37] Nan Wang, Sid Chi-Kin Chau, and Yue Zhou. 2021. Privacy-Preserving Energy Storage Sharing with Blockchain. In *ACM e-Energy 21'*.

[38] Jianing Zhang, Sid Chi-Kin Chau, and Minghua Chen. 2019. Stay or Switch: Competitive Online Algorithms for Energy Plan Selection in Energy Markets with Retail Choice. In *ACM e-Energy 19'*.

[39] Jing Zhao, Taeho Jung, Yu Wang, and Xiangyang Li. 2014. Achieving differential privacy of data disclosure in the smart grid. In *IEEE Infocom*.

[40] Shengru Zhou. 2017. *An Introduction to Retail Electricity Choice in the United States*. Technical Report. NREL.

[41] Yue Zhou and Sid Chi-Kin Chau. 2021. Sharing Economy Meets Energy Markets: Group Purchasing of Energy Plans in Retail Energy Markets. In *ACM BuildSys 21'*.

APPENDIX

A PROOFS OF RESULTS

Theorem 2. *If the connection fees and disconnection fees of all energy plans are identical, i.e.,* $\mathtt{c}_j = \mathtt{c}$ *and* $\mathtt{d}_j = \mathtt{d}$ *for all* $j \in \mathcal{E}$*, then*

$$\mathtt{Cost}_i[\hat{x}_i] \leq 3 \cdot \mathtt{Opt}_i^T[x^{*T}_i]$$

Proof. Since $\mathtt{c}_j = \mathtt{c}$ and $\mathtt{d}_j = \mathtt{d}$ for all $j \in \mathcal{E}$, the switching cost is a constant between all states. In this case, we can reduce the problem of n states to the one of 2 states only. When deciding the switching decision, one only needs to consider whether to stay with the current stay or change to another state that has a lower state cost (as the switching cost of any state is a constant). □

Theorem 3. *We consider two schemes of mutual compensations:*

(1) **Egalitarian Cost-sharing:**

$$\theta_i = C_i^t(\mathcal{E}\backslash\{\mathfrak{g}\}) + \frac{\sum_{i' \in X}\left(C_{i'}^t(\mathfrak{g}) - C_{i'}^t(\mathcal{E}\backslash\{\mathfrak{g}\})\right)}{|X|} - \mathtt{Opt}_i^t[\mathfrak{g}] \quad (17)$$

(2) **Proportional Cost-sharing:**

$$\theta_i = C_i^t(\mathcal{E}\backslash\{\mathfrak{g}\})\frac{\sum_{i' \in X} C_{i'}^t(\mathfrak{g})}{\sum_{i' \in X} C_{i'}^t(\mathcal{E}\backslash\{\mathfrak{g}\})} - \mathtt{Opt}_i^t[\mathfrak{g}] \quad (18)$$

Then $(\theta_i)_{i \in X}$ *in both schemes will satisfy individual rationality and budget balance, when group feasibility is satisfied.*

Proof. First, we consider egalitarian cost-sharing. Note that $\mathtt{Opt}_i^t[\mathfrak{g}] + \theta_i \geq 0$ for all $i \in X$. Suppose group feasibility is satisfied, then we obtain $\sum_{i \in X}\left(C_i^t(\mathfrak{g}) - C_i^t(\mathcal{E}\backslash\{\mathfrak{g}\})\right) < 0$. We show individual rationality as follows:

$$\mathtt{Opt}_i^t[\mathfrak{g}] + \theta_i = C_i^t(\mathcal{E}\backslash\{\mathfrak{g}\}) + \frac{\sum_{i' \in X}\left(C_{i'}^t(\mathfrak{g}) - C_{i'}^t(\mathcal{E}\backslash\{\mathfrak{g}\})\right)}{|X|} < C_i^t(\mathcal{E}\backslash\{\mathfrak{g}\})$$

Next, we show budget balance as follows:

$$\sum_{i\in X}\left(\mathtt{Opt}_i^t[\mathfrak{g}]+\theta_i\right)=\sum_{i\in X}\left(C_i^t(\mathcal{E}\backslash\{\mathfrak{g}\})+\frac{\sum_{i\in X}\left(C_i^t(\mathfrak{g})-C_i^t(\mathcal{E}\backslash\{\mathfrak{g}\})\right)}{|X|}\right)$$
$$=\sum_{i\in X}C_i^t(\mathcal{E}\backslash\{\mathfrak{g}\})+\sum_{i\in X}C_i^t(\mathfrak{g})-\sum_{i\in X}C_i^t(\mathcal{E}\backslash\{\mathfrak{g}\})=\sum_{i\in X}C_i^t(\mathfrak{g})$$

Similarly, we consider proportional cost-sharing. Suppose group feasibility is satisfied, then we obtain $\frac{\sum_{i\in X}C_i^t(\mathfrak{g})}{\sum_{i\in X}C_i^t(\mathcal{E}\backslash\{\mathfrak{g}\})}<1$. We show individual rationality as follows:

$$\mathtt{Opt}_i^t[\mathfrak{g}]+\theta_i=C_i^t(\mathcal{E}\backslash\{\mathfrak{g}\})\frac{\sum_{i'\in X}C_{i'}^t(\mathfrak{g})}{\sum_{i'\in X}C_{i'}^t(\mathcal{E}\backslash\{\mathfrak{g}\})}<C_{i'}^t(\mathcal{E}\backslash\{\mathfrak{g}\})$$

Next, we show budget balance as follows:

$$\sum_{i\in X}\left(\mathtt{Opt}_i^t[\mathfrak{g}]+\theta_i\right)=\sum_{i\in X}\left(C_i^t(\mathcal{E}\backslash\{\mathfrak{g}\})\frac{\sum_{i'\in X}C_{i'}^t(\mathfrak{g})}{\sum_{i'\in X}C_{i'}^t(\mathcal{E}\backslash\{\mathfrak{g}\})}\right)=\sum_{i\in X}C_i^t(\mathfrak{g})$$

□

B ZERO-KNOWLEDGE PROOFS

In this section, we provide a brief explanation of the concepts of Zero-Knowledge Proofs (ZKPs) in this section. More details can be found in a standard cryptography textbook (e.g., [8]).

First, we define some system parameters. Denote by $\mathbb{Z}_p=\{0,...,p-1\}$ a finite field of integers modulo p, for encrypting private data. For brevity, we simply write "$x+y$" and "$x\cdot y$" for modular arithmetic without explicitly mentioning"$\mathtt{mod}\ p$". We consider a usual finite group $\mathbb{G}$ of order p. We pick g,h as two generators of $\mathbb{G}$, such that they can generate every element in $\mathbb{G}$ by taking proper powers, namely, for each $e\in\mathbb{G}$, there exist $x,y\in\mathbb{Z}_p$ such that $e=g^x=h^y$. The classical discrete logarithmic assumption states that given g^x, it is computationally hard to obtain x, which underlies the security of many cryptosystems.

In a zero-knowledge proof (ZKP) (of knowledge), a prover convinces a verifier of the knowledge of a secret without revealing the secret. For example, to show the knowledge of $(x,\mathtt{r})$ for $\mathtt{Cm}(x,\mathtt{r})$ without revealing $(x,\mathtt{r})$. A zero-knowledge proof of knowledge should satisfy completeness (i.e., the prover always can convince the verifier if knowing the secret), soundness (i.e., the prover cannot convince a verifier if not knowing the secret) and zero-knowledge (i.e., the verifier cannot learn the secret).

B.1 Σ-Protocol

Σ-Protocol is a general approach to construct zero-knowledge proofs. Given a computationally non-invertible function $f(\cdot)$ that satisfies homomorphic property $f(a+b)=f(a)+f(b)$ and $f(x)=y$, one can prove the knowledge of the concealed x:

(1) First, the prover sends a commitment $y'=f(x')$, for a random x', to the verifier.
(2) Next, the verifier replies with a random challenge β.
(3) The prover replies with $z=x'+\beta\cdot x$ (which does not reveal x).
(4) Finally, the verifier checks whether $f(z)\stackrel{?}{=}y'+\beta\cdot y$.

B.2 Σ-Protocol Based Zero-knowledge Proofs

Next, we present several instances of zero-knowledge proofs based on Σ-protocol:

- **ZKP of Commitment** ($\mathtt{zkpCm}[x,\mathtt{Cm}(x)]$): Given $\mathtt{Cm}(x,r)$, a prover can convince a verifier of the knowledge of x without revealing (x,r). The corresponding protocol is described by $\Pi_{\mathtt{zkpCm}}$.
- **ZKP of Membership** ($\mathtt{zkpMbs}[x_i\in\mathcal{X}]$): Given a set $\mathcal{X}=\{x_1,...,x_n\}$ and $\mathtt{Cm}(x,\mathtt{r})$, a prover can convince a verifier of the knowledge of $x\in\mathcal{X}$ without revealing x. The corresponding protocol is described by $\Pi_{\mathtt{zkpMbs}}$.
- **ZKP of Non-Negativity** ($\mathtt{zkpNN}[x\geq 0]$): Given $\mathtt{Cm}(x,\mathtt{r})$, a prover can convince a verifier of the knowledge of $x\geq 0$ without revealing x. The corresponding protocol is described by $\Pi_{\mathtt{zkpNN}}$.

Algorithm 4 $\Pi_{\mathtt{zkpCm}}$: *Prove the knowledge of* $(x,\mathtt{r})$ *in a given commitment* $\mathtt{Cm}(x,\mathtt{r})$*, without revealing* $(x,\mathtt{r})$

Input: $\mathtt{Cm}(x,\mathtt{r})$ (known to the verifier and prover); $(x,\mathtt{r})$ (known to the prover only)
Output: Pass or Fail

1: The prover generates a pair of random numbers $(x',\mathtt{r}')\stackrel{\$}{\leftarrow}\mathbb{Z}_p^2$ and announces the commitment $\mathtt{Cm}(x',\mathtt{r}')$ to the verifier
2: The verifier generates a random number $\psi\stackrel{\$}{\leftarrow}\mathbb{Z}_p$ and announces ψ to the prover
3: The prover computes $z_x\leftarrow x'+\psi\cdot x$ and $z_\mathtt{r}\leftarrow\mathtt{r}'+\psi\cdot\mathtt{r}$, and announces $(z_x,z_\mathtt{r})$ to the verifier

▷ *The verifier checks the following*
4: **if** $g^{z_x}\cdot h^{z_\mathtt{r}}=\mathtt{Cm}(x',\mathtt{r}')\cdot\mathtt{Cm}(x,\mathtt{r})^\psi$ **then**
5: **return** Pass
6: **else**
7: **return** Fail
8: **end if**

▷ *The ZKP of commitment is* $\mathtt{zkpCm}[x,\mathtt{Cm}(x)]=\{\mathtt{Cm}(x,\mathtt{r});\mathtt{Cm}(x',\mathtt{r}'),z_x,z_\mathtt{r}\}$

Algorithm 5 $\Pi_{\mathtt{zkpMbs}}$: *Prove the knowledge that* $x_i\in\mathcal{X}\triangleq\{x_1,...,x_n\}$*, given commitment* $\mathtt{Cm}(x_i,\mathtt{r})$*, without revealing* x_i

Input: $\mathcal{X},\mathtt{Cm}(x_i,\mathtt{r})$ (known to the verifier and prover); $(x_i,\mathtt{r})$ (known to the prover only)
Output: Pass or Fail

1: The prover generates a pair of random numbers $(x'_j,\mathtt{r}'_j)\stackrel{\$}{\leftarrow}\mathbb{Z}_p^2$ and announces the commitments $\mathtt{Cm}(x'_j,\mathtt{r}'_j)$ for all $j\in\{1,...,n\}$ to the verifier
2: The prover generates a random number $\psi_j\stackrel{\$}{\leftarrow}\mathbb{Z}_p$ for each $j\in\{1,...,n\}\backslash\{i\}$, and computes

$$z_{x_j}\leftarrow\begin{cases}x'_j+\psi_j\cdot(x_i-x_j), & \text{if } j\in\{1,...,n\}\backslash\{i\}\\ x'_i, & \text{if } j=i\end{cases}$$

3: The verifier generates a random number $\psi\stackrel{\$}{\leftarrow}\mathbb{Z}_p$ and announces ψ to the prover
4: The prover sets $\psi_i\leftarrow\psi-\sum_{j\neq i}\psi_j$ and $z_{\mathtt{r}_j}\leftarrow\mathtt{r}'_j+\mathtt{r}\cdot\psi_j$ for all $j\in\{1,...,n\}$, and then announces $(\psi_j,z_{\mathtt{r}_j})_{j=1}^n$ to the verifier

▷ *The verifier checks the following*
5: **if** $g^{z_{x_j}}\cdot h^{z_{\mathtt{r}_j}}=\mathtt{Cm}(x'_j,\mathtt{r}'_j)\cdot\left(\frac{\mathtt{Cm}(x_i,\mathtt{r})}{g^{x_j}}\right)^{\psi_j}$ for all $j\in\{1,...,n\}$ and $\psi=\sum_{i=1}^n\psi_j$ **then**
6: **return** Pass
7: **else**
8: **return** Fail
9: **end if**

▷ *The ZKP of membership is* $\mathtt{zkpMbs}[x_i\in\mathcal{X}]=\{\mathtt{Cm}(x_i,\mathtt{r});\{\mathtt{Cm}(x'_j,\mathtt{r}'_j)\}_{j=1}^n,(\psi_j,z_{x_j},z_{\mathtt{r}_j})_{j=1}^n\}$

B.3 Non-interactive Zero-knowledge Proofs

An interactive zero-knowledge proof that requires a verifier-provided challenge can be converted to a non-interactive one by Fiat-Shamir heuristic to remove the verifier-provided challenge.

Let $\mathcal{H}(\cdot)\mapsto\mathbb{Z}_p$ be a cryptographic hash function. Given a list of commitments $(\mathtt{Cm}_1,...,\mathtt{Cm}_r)$, one can map to a single hash value

Algorithm 6 Π_{zkpNN}: *Prove the knowledge that* $x \geq 0$ *by showing there exist* $(b_1, ..., b_m)$ *such that* $b_i \in \{0, 1\}$ *for* $i \in \{0, ..., m\}$ *and* $\sum_{i=1}^{m} b_i \cdot 2^{i-1} = x$, *given commitment* $\mathtt{Cm}(x, \mathsf{r})$, *without revealing* x

Input: $\mathtt{Cm}(x, \mathsf{r})$ (known to the verifier and prover); (x, r) (known to the prover only)
Output: Pass or Fail

1: The prover announces $(\mathtt{Cm}(b_i, \mathsf{r}_i))_{i=1}^{m}$ to the verifier, and uses $\Pi_{\mathsf{zkpMbs}}[\{0,1\}, \mathtt{Cm}(b_i, \mathsf{r}_i), b_i]$ to prove that $b_i \in \{0, 1\}$
2: The prover generates a random number $\mathsf{r}' \xleftarrow{\$} \mathbb{Z}_p$ and announces the commitment $\mathtt{Cm}(0, \mathsf{r}')$ to the verifier
3: The verifier generates a random number $\psi \xleftarrow{\$} \mathbb{Z}_p$ and announces ψ to the prover
4: The prover computes $z_r \leftarrow \mathsf{r}' + \psi \cdot (\sum_{i=1}^{m} \mathsf{r}_i \cdot 2^{i-1} - \mathsf{r})$ and announces z_r to the verifier

▷ *The verifier checks the following*

5: **if** $h^{z_r} = \mathtt{Cm}(0, \mathsf{r}') \cdot \mathtt{Cm}(x, \mathsf{r})^{-\psi} \cdot \prod_{i=1}^{m} \mathtt{Cm}(b_i, \mathsf{r}_i)^{\psi \cdot 2^{i-1}}$ **then**
6: **return** Pass
7: **else**
8: **return** Fail
9: **end if**

▷ *The ZKP of non-negativity is* $\mathsf{zkpNN}[x \geq 0] = \{\mathtt{Cm}(x, \mathsf{r}), \{\mathtt{Cm}(b_i, \mathsf{r}_i)\}_{i=1}^{m}; \mathtt{Cm}(0, \mathsf{r}'), z_r\}$

by $\mathcal{H}(\mathtt{Cm}_1|...|\mathtt{Cm}_r)$, where the input is the concatenated string of $(\mathtt{Cm}_1, ..., \mathtt{Cm}_r)$. In a Σ-protocol, one can set the challenge by $\beta = \mathcal{H}(\mathtt{Cm}_1|...|\mathtt{Cm}_r)$, where $(\mathtt{Cm}_1, ..., \mathtt{Cm}_r)$ are all the commitments generated by the prover prior to the step of verifier-provided challenge (Step 2 of Σ-protocol). Hence, the prover does not wait for the verifier-provided random challenge, and instead generates the random challenge himself. The verifier will generate the same challenge following the same procedure for verification. We denote the non-interactive versions of the previous zero-knowledge proofs by `nzkpCm`, `nzkpSum`, `nzkpMbs`, `nzkpNN`, respectively.

C SPDZ PROTOCOL

In this section, we present a simplified version of SPDZ protocol for the clarity of exposition. The full version can be found in [17, 18].

There are three phases in SPDZ protocol: (1) pre-processing phase, (2) online phase, and (3) output and validation phase. We write $\langle x \rangle$ as a *secretly shared* number, meaning that there is a vector $(x_1, ..., x_n)$, such that each party i knows only x_i. To reveal the secretly shared number $\langle x \rangle$, each party i broadcasts x_i to other parties. Then each party can reconstruct $x = \sum_{i=1}^{n} x_i$. We write $\langle\langle x \rangle\rangle$ meaning that both $\langle x \rangle$ and the respective MAC $\langle \gamma(x) \rangle$ are secretly shared.

C.1 Online Phase

In the online phase, the parties can jointly compute an arithmetic circuit, consisting of additions and multiplications with secretly shared input numbers.

C.1.1 Addition.

Given secretly shared $\langle x \rangle$ and $\langle y \rangle$, and a public known constant c, the following operations can be attained by local computation at each party, and then the outcome can be assembled from the individual shares:

A1) $\langle x \rangle + \langle y \rangle$ can be computed by $(x_1 + y_1, ..., x_n + y_n)$.
A2) $c \cdot \langle x \rangle$ can be computed by $(c \cdot x_1, ..., c \cdot x_n)$.
A3) $c + \langle x \rangle$ can be computed by $(c + x_1, x_2, ..., x_n)$.

C.1.2 Multiplication.

Given secretly shared $\langle x \rangle$ and $\langle y \rangle$, computing the product $\langle x \rangle \cdot \langle y \rangle$ involves a given multiplication triple. A multiplication triple is defined by $(\langle a \rangle, \langle b \rangle, \langle c \rangle)$, where a, b are some unknown random numbers and $c = a \cdot b$, are three secretly shared numbers already distributed among the parties. The triple is assumed to be prepared in a pre-processing phase. To compute $\langle x \rangle \cdot \langle y \rangle$, it follows the below steps of operations (A4):

A4.1) Compute $\langle \epsilon \rangle = \langle x \rangle - \langle a \rangle$ (by A1). Then, reveal $\langle \epsilon \rangle$, which does not reveal x.
A4.2) Compute $\langle \delta \rangle = \langle y \rangle - \langle b \rangle$. Then, reveal $\langle \delta \rangle$.
A4.3) Finally, compute $\langle x \rangle \cdot \langle y \rangle = \langle c \rangle + \epsilon \cdot \langle b \rangle + \delta \cdot \langle a \rangle + \epsilon \cdot \delta$ (by A1-A3).

C.1.3 Message Authentication Code.

To safeguard against dishonest parties, who may perform incorrect computation, an information-theoretical message authentication code (MAC) can be used for verification. We write a MAC key as a global number $\widetilde{\alpha}$, which is unknown to the parties, and is secretly shared as $\langle \widetilde{\alpha} \rangle$. Every secretly shared number is encoded by a MAC as $\gamma(x) = \widetilde{\alpha} x$, which is secretly shared as $\langle \gamma(x) \rangle$. For each $\langle x \rangle$, each party i holds a tuple $(x_i, \gamma(x)_i)$ and $\widetilde{\alpha}_i$, where $x = \sum_{i=1}^{n} x_i$, $\widetilde{\alpha} = \sum_{i=1}^{n} \widetilde{\alpha}_i$ and $\gamma(x) = \widetilde{\alpha} x = \sum_{i=1}^{n} \gamma(x)_i$. If any party tries to modify his share x_i uncoordinatedly, then he also needs to modify $\gamma(x)_i$ accordingly. Otherwise, $\gamma(x)$ will be inconsistent. However, it is difficult to modify $\gamma(x)_i$ without coordination among the parties, such that $\widetilde{\alpha} x = \sum_{i=1}^{n} \gamma(x)_i$. Hence, it is possible to detect incorrect computation (possibly by dishonest parties) by checking the MAC.

To check the consistency of x, there is no need to reveal $\langle \widetilde{\alpha} \rangle$. One only needs to reveal $\langle x \rangle$, and then reveals $\widetilde{\alpha}_i - x \cdot \gamma(x)_i$ from each party i. One can check whether $\sum_{i=1}^{n} (\widetilde{\alpha}_i - x \cdot \gamma(x)_i) \stackrel{?}{=} 0$ for consistency. To prevent a dishonest party from modifying his share x_i after learning the other party's x_j. Each party needs to commit his share x_i before revealing x_i to others.

To maintain the consistency of MAC for operations A1-A4, the MAC needs to be updated accordingly as follows:

B1) $\langle x \rangle + \langle y \rangle$: Update MAC by $(\gamma(x)_1 + \gamma(y)_1, ..., \gamma(x)_n + \gamma(y)_n)$.
B2) $c \cdot \langle x \rangle$: Update MAC by $(c \cdot \gamma(x)_1, ..., c \cdot \gamma(x)_n)$.
B3) $c + \langle x \rangle$: Update MAC by $(c \cdot \alpha_1 + \gamma(x)_1, ..., c \cdot \alpha_n + \gamma(x)_n)$.
B4) $\langle x \rangle \cdot \langle y \rangle$: Update MAC at each individual step of A4.1-A4.3 accordingly by B1-B3.

The additions and multiplications of $\langle\langle x \rangle\rangle$ and $\langle\langle y \rangle\rangle$ follow A1-A4 and the MACs will be updated accordingly by B1-B4.

To verify the computation of a function, it only requires to check the MACs of the revealed values and the final outcome, which can be checked all efficiently together in a batch at the final stage by a technique called "random linear combination".

C.2 Pre-processing Phase

In the pre-processing phase, all parties need to prepare a collection of triplets $(\langle a \rangle, \langle b \rangle, \langle c \rangle)$ where $c = a \cdot b$, each for a required multiplication operation. Assume that the parties hold secretly shared numbers $a = \sum_{i=1}^{N} a_i$ and $b = \sum_{i=1}^{N} b_i$ (which has been generated by local random generation). Note that $a \cdot b = \sum_{i=1}^{N} a_i b_i + \sum_{i=1}^{N} \sum_{j=i" i \neq j}^{N} a_i b_j$. $a_i b_i$ can be computed locally. To distribute $a_i b_j$, one can use partial homomorphic cryptosystems, with encryption function $\mathtt{Enc}[\cdot]$ and decryption function $\mathtt{Dec}[\cdot]$ using party i's public and private

$(K_i^{\mathsf{p}}, K_i^{\mathsf{p}})$. First, party i sends $\mathsf{Enc}_{K_i^{\mathsf{p}}}[a_i]$ to party j, who responds by $C_i = b_j \mathsf{Enc}_{K_i^{\mathsf{p}}}[a_i] - \mathsf{Enc}_{K_i^{\mathsf{p}}}[\tilde{c}_j]$, where $\tilde{c}_j$ is a random share generated by party j and is encrypted by party i's public key K_i^{s}. Then party i can obtain $\tilde{c}_j = \mathsf{Dec}_{K_i^{\mathsf{s}}}[C_j]$. Hence, $a_i b_j = \tilde{c}_i + \tilde{c}_j$, which are secret shares $a_i b_j$. The above generation assumes honest parties. To prevent cheating by dishonest parties, one would need to use proper zero-knowledge proofs before secret sharing [17, 18].

To generate a random mask $\langle\!\langle r^i \rangle\!\rangle$, each party j needs to generate a random share r_j^i locally. Then the parties follow the similar procedure of triplet generation to compute the secretly shared product $\langle \gamma(r^i) \rangle$, where $\gamma(r^i) = \widetilde{\alpha} r^i$.

C.3 Output and Validation Phase

We describe random linear combination for batch checking. To check the MACs of a number of secretly shared numbers $\langle\!\langle x^1 \rangle\!\rangle, ..., \langle\!\langle x^m \rangle\!\rangle$ in a batch, first generate a set of random $(\mathsf{r}^1, ..., \mathsf{r}^m)$. then reveal $\langle\!\langle x^1 \rangle\!\rangle, ..., \langle\!\langle x^m \rangle\!\rangle$. Each party i computes $\sum_{j=1}^m \mathsf{r}^j (\widetilde{\alpha}_i - x^j \cdot \gamma(x^j)_i)$ and reveals it. All parties check whether $\sum_{i=1}^N \sum_{j=1}^m \mathsf{r}^j (\widetilde{\alpha}_i - x^j \cdot \gamma(x^j)_i) \stackrel{?}{=} 0$ for consistency in a batch checking.

C.4 SPDZ Protocol

We summarize the SPDZ protocol as follows:

(1) *Pre-processing Phase*: In this phase, a collection of shared random numbers will be constructed that can be used to mask the private input numbers. For each private input number of party i, there is a shared random number $\langle\!\langle r^i \rangle\!\rangle$, where r^i is revealed to party i only, but not to other parties. All parties also prepare a collection of triplets $(\langle\!\langle a \rangle\!\rangle, \langle\!\langle b \rangle\!\rangle, \langle\!\langle c \rangle\!\rangle)$ where $c = a \cdot b$, each for a required multiplication operation.

(2) *Online Phase*: To secretly shares a private input number x^i using $\langle\!\langle r^i \rangle\!\rangle$, without revealing x^i, it proceeds as follows:
   1) Party i computes and reveals $z^i = x^i - r^i$ to all parties.
   2) Every party sets $\langle\!\langle x^i \rangle\!\rangle \leftarrow z^i + \langle\!\langle r^i \rangle\!\rangle$.

   To compute an arithmetic circuit, implement the required additions or multiplications by A1-A4 and the MACs are updated accordingly by B1-B4.

(3) *Output and Validation Phase*: All MACs will be checked for all revealed numbers and the final output value. It can check all in a batch using random linear combination. If there is any inconsistency in the MACs, then abort.

Note that SPDZ cannot guarantee abort with fairness – dishonest parties may learn some partial values, even when the protocol aborts. However, this is a fundamental problem for any multi-party computation protocol with a dishonest majority, where dishonest parties are not identifiable when the computation is aborted.

D ADDITIONAL PROTOCOLS

D.1 Basic Protocols with SPDZ

In this section, we present several common protocols based on SPDZ as sub-routines:

(1) Π_{RandBit} is a protocol that generates a secretly shared random bit $\langle\!\langle b \rangle\!\rangle \in \{0, 1\}$ via SPDZ.

(2) Π_{RandPos} is a protocol that generates a secretly shared random positive number $\langle\!\langle R \rangle\!\rangle > 0$ via SPDZ.

(3) $\Pi_{\min}[\langle\!\langle y \rangle\!\rangle, \langle\!\langle z \rangle\!\rangle]$ is a protocol that computes, via SPDZ, secretly shared output $\langle\!\langle x \rangle\!\rangle$ for given secretly shared values $\langle\!\langle y \rangle\!\rangle, \langle\!\langle z \rangle\!\rangle$, such that $x \leftarrow \min\{y, z\}$.

(4) $\Pi_{<}[\langle\!\langle x \rangle\!\rangle, \langle\!\langle y \rangle\!\rangle]$ is a protocol that compares, via SPDZ, two secretly shared values $\langle\!\langle x \rangle\!\rangle$ and $\langle\!\langle y \rangle\!\rangle$ and outputs 1 if $\langle\!\langle x \rangle\!\rangle - \langle\!\langle y \rangle\!\rangle < 0$, and 0 otherwise, without revealing other information about x, y.

(5) $\Pi_{=}[\langle\!\langle x \rangle\!\rangle, \langle\!\langle y \rangle\!\rangle]$ is a protocol that compares, via SPDZ, two secretly shared values $\langle\!\langle x \rangle\!\rangle$ and $\langle\!\langle y \rangle\!\rangle$ and outputs 1 if $\langle\!\langle x \rangle\!\rangle - \langle\!\langle y \rangle\!\rangle = 0$, and 0 otherwise, without revealing other information about x, y.

Algorithm 7 Π_{RandBit}: *Generate a secretly shared random bit* $\langle\!\langle b \rangle\!\rangle \in \{0, 1\}$ *via SPDZ, without revealing* b

Output: $\langle\!\langle b \rangle\!\rangle$
for $i \in X$ **do**
  User i generates a random local bit $b_i \stackrel{\$}{\leftarrow} \{0, 1\}$
  Secretly share b_i as $\langle\!\langle b_i \rangle\!\rangle$ to all users
  Announce $\mathtt{Cm}(b_i)$ to all users
  Apply $\Pi_{\mathsf{dzkpCm}}[\mathtt{Cm}(b_i), \langle\!\langle b_i \rangle\!\rangle]$ to prove the knowledge of b_i
  User i provides $\mathtt{nzkpMbs}[\{0, 1\}, \mathtt{Cm}(b_i); b_i]$ to prove $\langle\!\langle b_i \rangle\!\rangle \in \{0, 1\}$
end for
Compute $\langle\!\langle b \rangle\!\rangle \leftarrow \bigotimes_{i \in X} \langle\!\langle b_i \rangle\!\rangle$ via SPDZ, where $\otimes$ is XOR operator:

$$\langle\!\langle b_i \rangle\!\rangle \otimes \langle\!\langle b_{i'} \rangle\!\rangle \triangleq 1 - (1 - \langle\!\langle b_i \rangle\!\rangle \cdot (1 - \langle\!\langle b_{i'} \rangle\!\rangle)) \cdot (1 - (1 - \langle\!\langle b_i \rangle\!\rangle) \cdot \langle\!\langle b_{i'} \rangle\!\rangle)$$

return $\langle\!\langle b \rangle\!\rangle$

Algorithm 8 Π_{RandPos}: *Generate a secretly shared random positive number* $\langle\!\langle R \rangle\!\rangle > 0$ *via SPDZ, without revealing* R

Output: $\langle\!\langle R \rangle\!\rangle$
for $i \in X$ **do**
  User i generates a random local positive number $R_i \stackrel{\$}{\leftarrow} \mathbb{Z}_q$
  Secretly share R_i as $\langle\!\langle R_i \rangle\!\rangle$ to all users
  Announce $\mathtt{Cm}(R_i)$ to all users
  Apply $\Pi_{\mathsf{dzkpCm}}[\mathtt{Cm}(R_i), \langle\!\langle R_i \rangle\!\rangle]$ to prove the knowledge of R_i
  User i provides $\mathtt{nzkpNN}[\mathtt{Cm}(R_i); R_i]$ to prove $\langle\!\langle R_i \rangle\!\rangle \geq 0$
end for
Compute $\langle\!\langle R \rangle\!\rangle \leftarrow \sum_{i \in X} \langle\!\langle R_i \rangle\!\rangle$ via SPDZ
return $\langle\!\langle R \rangle\!\rangle + 1$

Algorithm 9 Π_{Min}: *Compute* $\langle\!\langle z \rangle\!\rangle \leftarrow \min\{\langle\!\langle x \rangle\!\rangle, \langle\!\langle y \rangle\!\rangle\}$ *via SPDZ, in a privacy-preserving manner without revealing* (z, x, y)

Input: $\langle\!\langle x \rangle\!\rangle, \langle\!\langle y \rangle\!\rangle$ (already secretly shared among users)
Output: $\langle\!\langle z \rangle\!\rangle$
Generate a secretly shared random bit $\langle\!\langle b \rangle\!\rangle \leftarrow \Pi_{\mathsf{RandBit}}$
Randomly shuffle (x, y) based on b by the following via SPDZ:

$$\langle\!\langle u \rangle\!\rangle \leftarrow \langle\!\langle x \rangle\!\rangle + \langle\!\langle b \rangle\!\rangle \cdot (\langle\!\langle y \rangle\!\rangle - \langle\!\langle x \rangle\!\rangle)$$
$$\langle\!\langle v \rangle\!\rangle \leftarrow \langle\!\langle y \rangle\!\rangle + \langle\!\langle b \rangle\!\rangle \cdot (\langle\!\langle x \rangle\!\rangle - \langle\!\langle y \rangle\!\rangle)$$

Generate a secretly shared random positive number $\langle\!\langle R \rangle\!\rangle \leftarrow \Pi_{\mathsf{RandPos}}$
Compute $\langle\!\langle w \rangle\!\rangle \leftarrow \langle\!\langle R \rangle\!\rangle \cdot (\langle\!\langle u \rangle\!\rangle - \langle\!\langle v \rangle\!\rangle)$ via SPDZ
Open $\langle\!\langle w \rangle\!\rangle$ and its MAC to all users
if Checking MAC failed **then**
  Abort
end if
if $w \geq 0$ **then**
  Set $\langle\!\langle z \rangle\!\rangle \leftarrow \langle\!\langle v \rangle\!\rangle$
else
  Set $\langle\!\langle z \rangle\!\rangle \leftarrow \langle\!\langle u \rangle\!\rangle$
end if
return $\langle\!\langle z \rangle\!\rangle$

Algorithm 10 $\Pi_{<}$: *Output 1 for given* $\langle\langle x\rangle\rangle, \langle\langle y\rangle\rangle$, *if* $\langle\langle x\rangle\rangle > \langle\langle y\rangle\rangle$ *via SPDZ, in a privacy-preserving manner without revealing* (x, y)

Input: $\langle\langle x\rangle\rangle, \langle\langle y\rangle\rangle$ (already secretly shared among users)
Output: 1, if $\langle\langle x\rangle\rangle > \langle\langle y\rangle\rangle$. Otherwise, 0
Generate a secretly shared random positive number $\langle\langle R\rangle\rangle \leftarrow \Pi_{\mathsf{RandPos}}$
Compute $\langle\langle w\rangle\rangle \leftarrow \langle\langle R\rangle\rangle \cdot (\langle\langle x\rangle\rangle - \langle\langle y\rangle\rangle)$ via SPDZ
Open $\langle\langle w\rangle\rangle$ and its MAC to all users
if Checking MAC failed **then**
    Abort
end if
if $w > 0$ **then**
    return 1
else
    return 0
end if

Algorithm 11 $\Pi_{=}$: *Output 1 for given* $\langle\langle x\rangle\rangle, \langle\langle y\rangle\rangle$, *if* $\langle\langle x\rangle\rangle = \langle\langle y\rangle\rangle$ *via SPDZ, in a privacy-preserving manner without revealing* (x, y)

Input: $\langle\langle x\rangle\rangle, \langle\langle y\rangle\rangle$ (already secretly shared among users)
Output: 1, if $\langle\langle x\rangle\rangle = \langle\langle y\rangle\rangle$. Otherwise, 0
Generate a secretly shared random positive number $\langle\langle R\rangle\rangle \leftarrow \Pi_{\mathsf{RandPos}}$
Compute $\langle\langle w\rangle\rangle \leftarrow \langle\langle R\rangle\rangle \cdot (\langle\langle x\rangle\rangle - \langle\langle y\rangle\rangle)$ via SPDZ
Open $\langle\langle w\rangle\rangle$ and its MAC to all users
if Checking MAC failed **then**
    Abort
end if
if $w = 0$ **then**
    return 1
else
    return 0
end if

D.2 Privacy-preserving Payments on Blockchain

On Ethereum, one can create tokens on the ledger to represent certain digital assets. Our mutual compensation payment system is implemented by ERC20 tokens [21]. To make payment among each other, users are required to purchase tokens that will be subsequently transferred to each other and redeemed. By default, the transaction records on the ledger are completely visible to the public. In this section, we incorporate privacy protection to hide the transaction records on the ledger. As in other privacy-preserving blockchain platforms (e.g., Zether [9]), we conceal the balances and transaction values in the ledger by the respective cryptographic commitments instead of plaintext values. For example, $\mathtt{Cm}(\mathtt{Bal}(\mathsf{ad}_i))$ will be recorded as the balance for account ad_i on the ledger.

To initiate a transaction of tokens from ad_i to $\mathsf{ad}_{i'}$ with transaction value $\mathtt{val}$, a user submits a transaction request to the blockchain: $\mathtt{tx} = (\mathsf{ad}_i, \mathsf{ad}_{i'}, \mathtt{val})$, along with a signature $\mathtt{sign}_{K_i^{\mathsf{s}}}(\mathtt{tx})$ using the private key K_i^{s} associated with ad_i. To pay mutual compensations among multiple users, the users submit a *multi-transaction*, denoted by $\mathtt{mtx} = (\mathsf{ad}_i, \mathsf{ad}_{i'}, \mathtt{val}_i)_{i=1}^N$, which will be executed, only if $\mathtt{Bal}(\mathsf{ad}_i) \geq \mathtt{val}_i$ for all i and multi-signature $\mathtt{sign}_{(K_i^{\mathsf{s}})_{i=1}^N}(\mathtt{mtx})$ is present. To hide transfer amounts, a multi-transaction can be concealed as $\mathtt{mtx} = (\mathsf{ad}_i, \mathsf{ad}_{i'}, \mathtt{Cm}(\mathtt{val}_i))_{i=1}^N$. Since Pedersen commitment satisfies homomorphic property, the concealed transaction value can be added to the concealed balance as follows:

$$\mathtt{Cm}(\mathtt{Bal}(\mathsf{ad}_i)) \leftarrow \mathtt{Cm}(\mathtt{Bal}(\mathsf{ad}_i)) \cdot \mathtt{Cm}(\mathtt{val}_i)$$

However, the transaction may be invalid, when $\mathtt{Bal}(\mathsf{ad}_i) \leq \mathtt{val}_i$. Hence, each user must provide $\mathtt{nzkpNN}[\mathtt{Bal}(\mathsf{ad}_i) - \mathtt{val}_i \geq 0]$ along with each transaction request to prove the non-negativity of the resultant balance. Otherwise, the transaction request will be denied.

In Stage 4, each user i pays the net mutual compensation amount ϕ_i (which may be negative, if the user receives compensation). Note that it is evident that $\sum_{i\in X} \phi_i = 0$ (namely, the sum of net mutual compensations should be zero). Hence, it suffices to consider the scenario that all users transfer the payments to a dummy address $\mathsf{ad}_{\mathsf{dummy}}$ with zero total transaction value, with each transferring a transaction value of $\mathtt{Cm}(\phi_i)$ in a commitment. To prove the validity of the multi-transaction, the users need to provide a ZKP of summation, such that the summation of all transferred amounts equals zero, namely, $\prod_{i\in X} \mathtt{Cm}(\phi_i) = \mathtt{Cm}(0)$. This can be constructed by Σ-protocol. However, we need a distributed version of ZKP of summation, as it can be validated by all users. A distributed ZKP of summation via SPDZ is described in Π_{dzkpsum} (Algorithm 12).

Algorithm 12 Π_{dzkpSum}: *Prove the knowledge of* $y = \sum_{i=1}^n x_i$ *in given commitments* $\left(\mathtt{Cm}(x_1, \mathsf{r}_1),\ldots,\mathtt{Cm}(x_n, \mathsf{r}_n)\right)$ *and public known value* y, *via SPDZ*

Input: $\left(\mathtt{Cm}(x_1, \mathsf{r}_1),\ldots,\mathtt{Cm}(x_n, \mathsf{r}_n)\right)$ (known to the verifier and prover)
Output: Pass or Fail
Each user i announces $\mathtt{Cm}(x_i, \mathsf{r}_i)$ and secretly shares $\langle\langle \mathsf{r}_i \rangle\rangle$ with all users
Each user i randomly generates $\mathsf{r}'_i \leftarrow \mathbb{Z}_p$ and secretly shares $\langle\langle \mathsf{r}'_i \rangle\rangle$ before announcing $\mathtt{Cm}(0, \mathsf{r}'_i)$ to all users
All users compute $C' \leftarrow \prod_{i=1}^N \mathtt{Cm}(0, \mathsf{r}'_i)$ and obtain a random challenge $\beta \leftarrow \mathcal{H}(C')$
All users compute $\langle\langle z_r \rangle\rangle \leftarrow \sum_{i=1}^n \langle\langle \mathsf{r}'_i \rangle\rangle + \beta \cdot \sum_{i=1}^n \langle\langle \mathsf{r}_i \rangle\rangle$ via SPDZ
Reveal $\langle\langle z_r \rangle\rangle$ and its MAC to all users

▷ *All users check the following*

if $g^{\beta \cdot y} \cdot h^{z_r} \stackrel{?}{=} C' \cdot \prod_{i=1}^n \mathtt{Cm}(x_i, \mathsf{r}_i)^{\beta}$ and checking MAC passed **then**
    return Pass
else
    return Fail
end if

▷ *The ZKP of summation is* $\mathtt{zkpSum}[\sum_{i=1}^n x_i = y] = \left\{\left(\mathtt{Cm}(x_1, \mathsf{r}_1),\ldots,\mathtt{Cm}(x_n, \mathsf{r}_n)\right); C', z_r\right\}$

Next, we describe the process of privacy-preserving payments of net mutual payments as follows:

(1) Each user i retains secretly-shared $\left(\langle\langle \phi_i \rangle\rangle\right)_{i\in X}$ in Stage 3, and announces its commitment $\mathtt{Cm}(\phi_i, \mathsf{r}_i)$ to all users, and then secretly shares $\langle\langle \mathsf{r}_i \rangle\rangle$ via SPDZ.
(2) All users generate ZKP of summation by Π_{dzkpsum} to prove that $\sum_{i\in X} \phi_i = 0$.
(3) Each user i generates $\mathtt{nzkpNN}[\mathtt{Bal}(\mathsf{ad}_i) - \phi_i \geq 0]$ based on $\mathtt{Cm}(\phi_i, \mathsf{r}_i)$.
(4) The users submit a multi-transaction request

$$\mathtt{mtx} = \left(\mathsf{ad}_i, \mathsf{ad}_{\mathsf{dummy}}, \mathtt{Cm}(\phi_i, \mathsf{r}_i)\right)_{i\in X}$$

to the blockchain ledger, along with the following ZKPs:

$$\mathtt{nzkpSum}[\sum_{i\in X} \phi_i = 0] \text{ and } \mathtt{nzkpNN}[\mathtt{Bal}(\mathsf{ad}_i) - \phi_i \geq 0]_{i\in X}$$

(5) The blockchain ledger verifies $\mathtt{nzkpSum}$ and $\mathtt{nzkpNN}$ before recording the transaction.