energies

MDPI

Article

Privacy-Preserving Peer-to-Peer Energy Trading in Blockchain-Enabled Smart Grids Using Functional Encryption

Ye-Byoul Son, Jong-Hyuk Im, Hee-Yong Kwon, Seong-Yun Jeon and Mun-Kyu Lee *

Department of Computer Engineering, Inha University, Incheon 22212, Korea; byoul0114@gmail.com (Y.-B.S.); imjhyuk@gmail.com (J.-H.I.); heeyong.kr@gmail.com (H.-Y.K.); roland.korea@gmail.com (S.-Y.J.)

* Correspondence: mklee@inha.ac.kr; Tel.: +82-32-860-7456

Received: 31 January 2020; Accepted: 7 March 2020; Published: 12 March 2020

Abstract: Advanced smart grid technologies enable energy prosumers to trade surplus energy from their distributed renewable energy sources with other peer prosumers through peer-to-peer (P2P) energy trading. In many previous works, P2P energy trading was facilitated by blockchain technology through blockchain's distributive nature and capacity to run smart contracts. However, the feature that all the data and transactions on a blockchain are visible to all blockchain nodes may significantly threaten the privacy of the parties participating in P2P energy trading. There are many previous works that have attempted to mitigate this problem. However, all these works focused on the anonymity of participants but did not protect the data and transactions. To address this issue, we propose a P2P energy trading system on a blockchain where all bids are encrypted and peer matching is performed on the encrypted bids by a functional encryption-based smart contract. The system guarantees that the information encoded in the encrypted bids is protected, but the peer matching transactions are performed by the nodes in a publicly verifiable manner through smart contracts. We verify the feasibility of the proposed system by implementing a prototype composed of smart meters, a distribution system operator (DSO) server, and private Ethereum blockchain.

Keywords: smart grid; blockchain; peer-to-peer energy trading; privacy protection; functional encryption

1. Introduction

Distributed renewable energy sources, such as solar panels and wind turbines, are drastically changing the way electricity is generated and consumed [1,2]. Energy *prosumers*, i.e., energy consumers producing their own electrical power, are rapidly proliferating. Advanced smart grid technologies enable these prosumers to trade surplus energy with other peer prosumers through peer-to-peer (P2P) energy trading [3]. P2P energy trading is expected to be one of the most important elements of next-generation power systems [1]. It can provide numerous benefits, such as increasing the overall efficiency of power systems and reducing power outages. For customers, it may create a competitive energy market and allow access to alternative energy sources according to their individual preferences [1].

In many ongoing trials, P2P energy trading is being facilitated by blockchain technology through blockchain's distributive nature and capacity to run smart contracts [3,4]. Such trading systems cannot be fully independent of the existing infrastructure, but the integration of new systems into conventional centralized energy systems is crucial [4]. An electric power system requires two fundamental components: the physical system for electricity transmission and the information system for settlement, balancing, and billing. In the P2P trading context, a blockchain can provide a partial solution for the latter, such as matching and contracting between sellers and buyers and recording

all transactions in an immutable way. However, the physical functionality must be provided by the existing infrastructure. For example, even when two peers agree on an energy trade between them, there is no way to physically exchange energy if they are away from each other but not mobile. This situation is entirely different from data transmission in a communication system because power routing in dynamic routing configurations is very challenging in comparison with data-packet routing [1]. Therefore, prosumers must use the power-distribution lines connected to an existing utility company. Furthermore, considering that most renewable energy sources are intermittent, prosumers may still want the stable power source from the utility company, as well as the renewable sources.

In this situation, a utility company serves as both a distribution system operator (DSO), renting transmission and distribution lines to prosumers for P2P trading, and a load serving entity (LSE) (i.e., a power retailer providing stable electricity in a traditional way). According to a recent survey conducted in Reference [3], prosumers preferred established authorities, such as energy suppliers and the local council, to less stable parties, such as blockchain energy start-ups, as P2P scheme organizers because P2P trading between prosumers requires numerous safeguards. In addition, although the type of technology running the scheme was not important to the prosumers, the mention of Bitcoin [5] caused a significant negative reaction, which was likely due to its extreme price instability [3]. Therefore, it would be desirable that a means for financially settling the energy transactions is established outside a blockchain. A desirable feature of this approach is that a prosumer's credit for produced energy and debit for consumed energy through P2P trading can be managed together with the energy consumption from the legacy line on a single billing account registered to the utility company, while a prosumer can remain anonymous to other prosumers. Nevertheless, a blockchain is still an attractive platform for peer matching and trade negotiations as blockchain transactions are transparent and cannot be manipulated even by the utility company. We remark that transparency was valued as the most important characteristic of a local energy trading network in another recent survey [6]. However, using a blockchain in this way can pose a potential threat to the security of the utility company. The blockchain will contain all the details of the transactions mediated by the company, which can reveal a significant portion of the trade secrets of company, such as the size and pattern of transactions and the company revenue through mediation fees. Even if a private/permissioned blockchain is used, as in many previous works in relevant literature [3,4,7], every node participating in the blockchain will learn confidential information about the company.

To address these issues, we aimed at achieving two main goals that, at a first glance, may seem to conflict with each other: transparency of the peer matching procedure and confidentiality of the matching information. For this purpose, we proposed a P2P energy trading system on a private Ethereum blockchain, where all bids are encrypted and peer matching is performed on the encrypted bids by a functional encryption (FE)-based smart contract. The system guarantees that the information encoded in the encrypted bids is protected from blockchain nodes, although the peer matching transactions are performed by the nodes in a publicly verifiable manner through smart contracts. The prosumers remain anonymous to each other, whereas the utility company knows their identity for accounting and billing purposes. Moreover, matching peers cannot repudiate their bids after the matching is complete. Accounting and billing are settled outside the blockchain. We verify the feasibility of the proposed system by implementing a prototype composed of smart meters, a DSO server, and a private Ethereum blockchain. According to the experimental results, the entire procedure for encrypting a bid, uploading this bid to the blockchain, and matching peers on the blockchain were completed in real time (i.e., in a few seconds). A part of this paper appeared at [8].

2. Preliminaries

2.1. Blockchain and Privacy

A blockchain is a distributed ledger, the integrity of which is ensured through interlinked, cryptographically signed, and time-stamped blocks that contain transactions [2,9]. The data and

transactions in blocks are validated by all participants ("nodes") based on a well-defined consensus protocol without a trusted third party. Although the blockchain concept originated from Bitcoin [5], the script language for Bitcoin is too limited to perform general transactions because it is designed mainly for a single-purpose application (i.e., exchanging Bitcoin cryptocurrency). For example, Bitcoin does not support the execution of loops or complex programs. Ethereum [10] is the most well-known solution of this problem. The Ethereum protocol supports a Turing-complete programming language and enables users to create virtually any form of transaction or application using smart contracts [11,12]. Smart contracts are computer programs that securely reside on the blockchain and automatically execute the specified function [1,13]. An Ethereum smart contract can be written in the Solidity language and is created by sending a contract-creation transaction to the blockchain network. Smart contracts are run on the Ethereum Virtual Machine (EVM), which is maintained by Ethereum nodes in a decentralized and distributed manner [14]. Because Ethereum supports complex applications, EVM requires a method of validating the smart contract. To prevent poorly written code, such as an infinite loop, Solidity uses 'gas' as the fundamental unit of computation when running a smart contract [15]. If the smart contract requires more gas than the gas limit defined at the time of its deployment, its execution is terminated, thereby avoiding any waste of resources.

Blockchain has the potential to fulfill various requirements for P2P applications; however, it faces a confidentiality issue [16,17]. Because the design philosophy of a blockchain is to provide integrity through publicly verifiable transactions, all the data and transactions on the blockchain are visible to all the participating nodes. Most blockchains, including Ethereum, provide simple anonymity by generating pseudorandom addresses instead of permanent user IDs. However, the transactions of the same user can be linked through the address, although his/her real ID is not revealed. There is a well-known attack wherein an attacker can trace the transmission history to match a pseudorandom address with a person holding a coin. To solve this problem, mixing services were designed to exchange a user's coin with those of other users randomly, so that the original owner of the coin is not identified. Mixcoin [18] and Coinshuffle [19] are examples of such services. Another approach is to provide zero-knowledge proof on the blockchain, such as zk-SNARK [20] and BulletProof [21]. Zerocash [22] and Zether [23] are well-known applications of zero-knowledge proof. However, Zerocash [22] and Zether [23], as well as mixing services, are limited to the confidential transfer of cryptocurrency, i.e., ensuring user anonymity when cryptocurrency is transferred. Although zk-SNARK [20] and BulletProof [21] can have a wider range of applications, they are not designed for directly performing confidential transactions, but for confidentially verifying the validity of a transaction. The challenge encountered when implementing a confidential transaction is that users must disclose their data to all blockchain nodes to perform transactions over the blockchain.

Therefore, a method of performing transactions (i.e., smart contracts) is required without opening the data to the nodes. Hawk [24] is a framework that builds privacy-preserving smart contracts. It effectively solved the issue discussed earlier, but by introducing a third party called the manager. The manager must be involved in every confidential transaction, thus causing transaction execution to require many interactions between the users and the manager.

2.2. Peer-to-Peer Energy Trading Through Blockchain

Recently, the energy business has been considering decentralized systems owing to the necessity of energy-sector decentralization and the fact that renewable energy is not suitable for the centralized legacy system [25,26]. Blockchain meets these demands; thus, there have been many trials of P2P energy trading based on blockchain technologies. Power Ledger [27] and LO3 Energy's Brooklyn Microgrid [28,29] are the best-known projects that use blockchains. Power Ledger is a P2P energy trading platform that uses blockchains; it was developed in Australia and has expanded its territory to include India, Japan, and other countries. Brooklyn Microgrid is an energy marketplace for the residents of Brooklyn, New York. There are other platforms, such as Slock.it [30] and SolarCoin [31], but they do not address the privacy issue. Additionally, there are P2P energy trading systems for

specific purposes, such as electric vehicle charging [32] and campus-scale energy auction [33]. However, the privacy issue was not addressed effectively in these systems, either.

To tackle the privacy threat against P2P energy trading, various approaches have been proposed. Li et al. [34] and Kang et al. [35] provided pseudonym-based solutions for P2P energy trading on consortium-based blockchains. Aitzhan and Svetinovic [7] presented a token-based energy trading system to enable peers to perform anonymous transactions on a Bitcoin-based blockchain. This can also be categorized as a pseudonym-based approach, where only the identities of peers are protected, but the transaction data are visible to all blockchain nodes. Gai et al. [36] proposed to prevent linking attacks and malicious data mining by adding noise to the trading distribution, which is similar to the well-known smart meter data obfuscation techniques [37–39] to prevent non-intrusive appliance load monitoring (NIALM) [40]. To be precise, the privacy-preserving mechanism in Reference [36] was to achieve differential privacy by creating dummy accounts and dividing accounts. Therefore, only the statistical information for each peer is protected, but the transaction data are not protected. To the best of our knowledge, there has been no proposal to protect transaction data while performing prosumer matching on a blockchain.

Finally, blockchains can be used for purposes different from trade negotiation. For example, Munsing et al. [41] proposed the use of a blockchain to decentralize the optimization of energy resources. Luo et al. [42] proposed to use a separate agent system for negotiation and a blockchain only as a transaction settlement mechanism.

2.3. Function-Hiding Inner Product Functional Encryption

Functional encryption (FE) is a special type of encryption scheme that supports operations on encrypted data [43–45]. In this section, we will explain only the special FE scheme, i.e., function-hiding inner product encryption (FHIPE) [46–50], although the original FE can provide a wider range of functions and properties. Let $\mathbf{x}$ and $\mathbf{y}$ be two vectors, and let $\mathsf{E}(\mathbf{x})$ and $\mathsf{E}(\mathbf{y})$ be their encryption. Roughly speaking, the decryption operation takes two ciphertexts $\mathsf{E}(\mathbf{x})$ and $\mathsf{E}(\mathbf{y})$ as inputs, and produces the inner product of $\mathbf{x}$ and $\mathbf{y}$, which is denoted as $\langle \mathbf{x}, \mathbf{y} \rangle$. This operation can be performed by any party but reveals no information about either $\mathbf{x}$ or $\mathbf{y}$, except $\langle \mathbf{x}, \mathbf{y} \rangle$.

In this study, we use the practical FHIPE scheme, Π_{IPE}, proposed by Kim et al. in 2018 [50]. This scheme uses pairing-based asymmetric bilinear groups. Therefore, we begin the description of Π_{IPE} with some definitions and properties of pairings. Let $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$ be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are two distinct groups of prime order q. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a mapping that maps two group elements from $\mathbb{G}_1$ and $\mathbb{G}_2$ onto a target group $\mathbb{G}_T$. We write the group operations in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ multiplicatively. Let the tuple $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, e)$ be an asymmetric bilinear group that satisfies the following properties:

- Both the bilinear map e and the group operations in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ can be computed efficiently.
- Map e satisfies $e(P, Q) \neq 1$, i.e., e has non-degeneracy.
- Map e is bilinear for all $x, y \in \mathbb{Z}_q$. In other words, map e satisfies $e(P^x, Q^y) = e(P, Q)^{xy}$.

For a group element $P \in \mathbb{G}_1$ and a row vector $\mathbf{v} = (v_1, \ldots, v_n)$, $P^{\mathbf{v}}$ denotes the vector of group elements $(P^{v_1}, \ldots, P^{v_n})$ as in Reference [50]. The bilinear map over groups is extended to vectors as follows:

$$e(P^{\mathbf{v}}, Q^{\mathbf{w}}) = \prod_{i=1,\ldots,n} e(P^{v_i}, Q^{w_i}) = e(P, Q)^{\langle \mathbf{v}, \mathbf{w} \rangle}, \quad (1)$$

where $\mathbf{v} = (v_1, \ldots, v_n)$ and $\mathbf{w} = (w_1, \ldots, w_n)$.

Now, we are ready to explain Π_{IPE}. Many inner product encryption schemes typically consist of four probabilistic polynomial time (PPT) algorithms: setup, key generation, encryption, and decryption [46–50]. However, it is more intuitive to use the notations, left and right encryptions, instead of key generation and encryption, respectively. This notation was already used in Reference [50]. Then, the Π_{IPE} scheme is defined with four PPT algorithms, Setup, LeftEncrypt, RightEncrypt, and Decrypt, as follows:

- $\mathsf{Setup}(1^\lambda) \rightarrow (\mathsf{op}, \mathsf{sk})$: Given a security parameter λ, the setup algorithm Setup outputs the public parameters op and the secret key sk corresponding to λ. Concretely, Setup samples an asymmetric bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, e)$, and chooses generators $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$. Next, the algorithm samples the matrix $\mathbf{B} \leftarrow \mathbb{GL}_n(\mathbb{Z}_q)$, where $\mathbb{GL}_n(\mathbb{Z}_q)$ is the general linear group of $(n \times n)$ matrices over $\mathbb{Z}_q$. Then, the algorithm sets the matrix $\mathbf{B}^* = \det(\mathbf{B}) \cdot (\mathbf{B}^{-1})^\top$ using $\det(\mathbf{B})$, the determinant of $\mathbf{B}$, and $(\mathbf{B}^{-1})^\top$, the transpose matrix of $\mathbf{B}^{-1}$. Finally, the setup algorithm outputs the public parameters $\mathsf{op} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, e)$ and the secret key $\mathsf{sk} = (\mathsf{op}, P, Q, \mathbf{B}, \mathbf{B}^*)$.
- $\mathsf{LeftEncrypt}(\mathsf{sk}, \alpha, \mathbf{x}) \rightarrow E_L(\mathbf{x})$: Given the secret key sk, a uniformly random element $\alpha \in \mathbb{Z}_q$, and a row vector $\mathbf{x} = (x_1, \ldots, x_n)$, the left encryption algorithm LeftEncrypt outputs a left encryption:
$$E_L(\mathbf{x}) = (L_1, L_2) = (P^{\alpha \cdot \det(\mathbf{B})}, P^{\alpha \cdot \mathbf{x} \cdot \mathbf{B}}).$$
- $\mathsf{RightEncrypt}(\mathsf{sk}, \beta, \mathbf{y}) \rightarrow E_R(\mathbf{y})$: Given the secret key sk, a uniformly random element $\beta \in \mathbb{Z}_q$, and a row vector $\mathbf{y} = (y_1, \ldots, y_n)$, the right encryption algorithm RightEncrypt outputs a right encryption:
$$E_R(\mathbf{y}) = (R_1, R_2) = (Q^{\beta}, Q^{\beta \cdot \mathbf{y} \cdot \mathbf{B}^*}).$$
- $\mathsf{Decrypt}(\mathsf{op}, E_L(\mathbf{x}), E_R(\mathbf{y})) \rightarrow z$: Given the public parameters op, the left encryption $E_L(\mathbf{x}) = (L_1, L_2)$, and the right encryption $E_R(\mathbf{y}) = (R_1, R_2)$, the decryption algorithm Decrypt computes:
$$D_1 = e(L_1, R_1) \text{ and } D_2 = e(L_2, R_2).$$
Then, it checks whether there is a z satisfying $(D_1)^z = D_2$ by solving a discrete logarithm problem [51] and either outputs z or a symbol implying that a valid z cannot be found. If there is such a z, then $z = \langle \mathbf{x}, \mathbf{y} \rangle$.

3. Energy Trading System Model

3.1. System Components

We consider a microgrid system consisting of prosumers, smart meters, an energy storage, a DSO, and a blockchain. Figure 1 illustrates the proposed system model. Each smart meter is connected to a prosumer that we simply call a user. According to the convention in relevant literature [7,52], we assume that a smart meter is a sealed tamper-proof device, i.e., even the user of the smart meter cannot extract or inject secret keying material, although the user initiates the power trading transactions by providing his/her smart meter with the desired amount and price of power to be sold or bought. Therefore, we assume that a smart meter never performs a malicious function, i.e., it is a trusted party. In addition, we consider a smart meter as not only a metering device but also a network node that provides the user with a connection to the outside world, such as the blockchain. However, it is also possible to set up a separate device for this purpose, if necessary. An energy storage is an energy pool equipped with a bi-directional communication flow. It is connected with each smart meter through a local distribution network. It is also connected to a DSO and a transmission system operator (TSO) that transmits electrical power from power plants through a transmission line. We assume that the energy storage is also a trusted party. In this paper, we do not deal with the details of power-transmission lines involving TSOs and power plants. A DSO is responsible for both the uni-directional power distribution from the transmission line and the bi-directional distribution between the prosumers via the energy storage. For the latter P2P energy trading, the DSO becomes a mediator and updates the seller credit and the buyer debit according to the seller-buyer matching information provided by the blockchain. The DSO also serves as an LSE, and each user has an account registered to the DSO. Thus, the DSO periodically charges each account associated with a user at a rate decided by (usage of energy from power plants) − (energy sale credit) + (energy purchase debit) + (fees for transaction mediation and distribution). The fees are not only for mediating P2P trading but also for the transmission and distribution of power from power plants, as in a traditional energy

service. The blockchain provides the peer users with a platform for energy trading where users can anonymously negotiate energy prices without revealing their identity or prices to any third party except the DSO. The blockchain implements this functionality by performing operations on users' encrypted bids with functional encryption. Finally, we assume that all data communication lines are protected by a proper traditional end-to-end encryption mechanism.

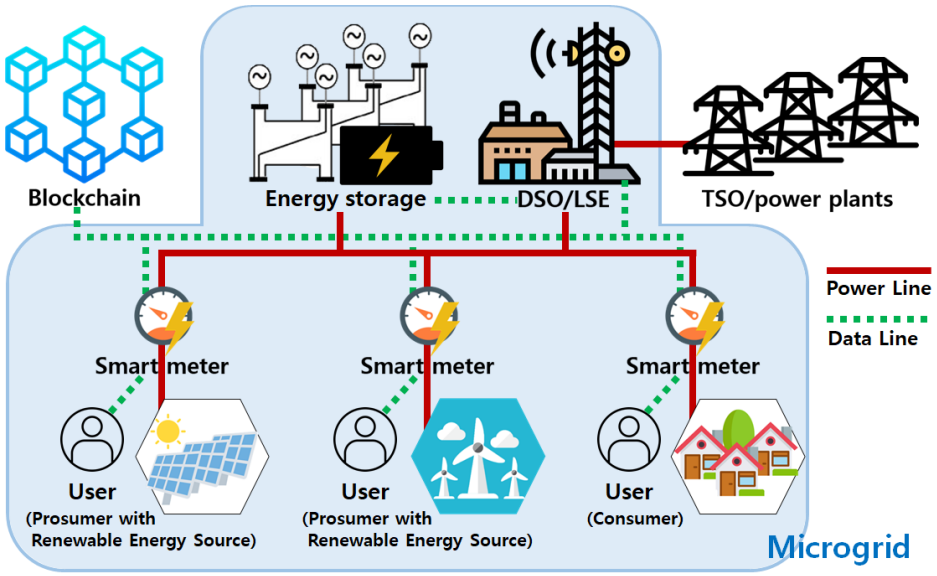

Figure 1. Proposed blockchain-enabled microgrid system model. DSO = distribution system operator; LSE = load serving entity; TSO = transmission system operator.

3.2. Threat Model and Security Goals

Based on the above system model, we assume the following threat model.

- The **blockchain** never deviates from the protocol specified by smart contracts. In addition, the integrity of the data written on the blockchain is guaranteed by the nature of the blockchain. However, each node participating in the blockchain network can see all the details of all the transactions and their related data if they are not encrypted. Therefore, from the viewpoint of the DSO and the users, the blockchain can be considered an honest-but-curious adversary that aims at obtaining any useful information from the user transactions but implements the protocol honestly and correctly.
- When mediating an energy transaction, the **DSO** may try to manipulate either the energy price or trading amount between the matched seller and buyer artificially to maximize its profit. Let us assume that the desired prices of a seller and a buyer are pp_S and pp_B, respectively. The matching for energy trading is successful when $pp_S \leq pp_B$. If $pp_S = pp_B$, the transactions are performed with this matched price. In contrast, if $pp_S < pp_B$, the price is settled according to a predefined policy, e.g., $(pp_S + pp_B)/2$. However, abusing the property that the users cannot see the other party's encrypted bid, the DSO may try to buy the energy for pp_S and sell it for pp_B. Second, assume that the desired trading amounts of a seller and a buyer are pa_S and pa_B, respectively. Then, the matched trading amount must be $min(pa_S, pa_B)$. However, if the fee for mediating P2P trading is not profitable enough, in comparison with that for legacy transmission and distribution, the DSO may attempt to intentionally reduce the volume of P2P transactions.
- **Users** may attempt to repudiate the amount and price of power that they had declared when initiating a bid.

To address the above threats, a P2P energy trading system must satisfy the following goals.

- **User Privacy**: The identities of users participating in P2P transactions must be kept private from blockchain nodes. The peers that conduct power transactions must not learn each other's identities. The identity of a user must not be linkable through multiple transactions involving this user.
- **DSO Privacy**: Because the DSO's profit depends on the fees for transaction mediation, the statistics on the transactions mediated by the DSO may reveal a significant portion of the company's trade secrets. Therefore, the contents of each bid, such as the amount and price of power, must be kept private from blockchain nodes.
- **Verifiability of transactions** and **integrity of bids**: Let **S** and **B** be the sets of sellers and buyers that uploaded bids on the blockchain, respectively. Let pp_U and pa_U be the price and amount of power declared by a user $U \in \mathbf{S} \cup \mathbf{B}$, respectively. If $\min_{S \in \mathbf{S}} pp_S \leq \max_{B \in \mathbf{B}} pp_B$, it must be guaranteed that the seller with the minimum bid, i.e., $S_m = \text{argmin}_{S \in \mathbf{S}} pp_S$, and the buyer with the maximum bid, i.e., $B_M = \text{argmax}_{B \in \mathbf{B}} pp_B$, will be matched for the P2P transaction. All operations must be verifiable by any participating user. Furthermore, the DSO must not be able to modify pp_{S_m}, pa_{S_m}, pp_{B_M}, and pa_{B_M} when mediating the transaction.
- **Nonrepudiation of bids**: The matched seller and buyer must not be able to repudiate their bids, i.e., S_m cannot deny pp_{S_m} and pa_{S_m}, and B_M cannot deny pp_{B_M} and pa_{B_M}.

4. Proposed Energy Trading System

In this section, we first explain the proposed algorithm that matches a seller and a buyer in a privacy-preserving manner using functional encryption. Then, we combine this algorithm with the system model explained in the previous section to build a prototype energy trading system.

4.1. Privacy-Preserving Matching Algorithm

First, we design a matching strategy that matches a seller and buyer pair for energy trading. Next, we extend this strategy to a privacy-preserving matching algorithm. As in the previous section, let **S** and **B** be the sets of sellers and buyers who want to trade energy, respectively. Let pp_U be the power price declared by a user $U \in \mathbf{S} \cup \mathbf{B}$. As in a typical matching strategy for various trading markets and auctions, e.g., a stock market, we will match the seller with the minimum bid, i.e., $S_m = \text{argmin}_{S \in \mathbf{S}} pp_S$, to the buyer with the maximum bid, i.e., $B_M = \text{argmax}_{B \in \mathbf{B}} pp_B$. Therefore, matching is not possible if $\min_{S \in \mathbf{S}} pp_S > \max_{B \in \mathbf{B}} pp_B$. To identify S_m and B_M easily, we maintain two array-based heap data structures: a min-heap H_S for sellers and a max-heap H_B for buyers, where the primary keys are the bid values. Therefore, S_m and B_M can be found at the roots of H_S and H_B, respectively. The insertion and deletion of a bid can be completed in either $O(\log_2 |\mathbf{S}|)$ or $O(\log_2 |\mathbf{B}|)$ time. After S_m and B_M are matched, the amount of power to be traded is decided as $min(pa_{S_m}, pa_{B_M})$, where pa_U is the power amount declared in user U's bid.

To perform the above-mentioned matching and heap updating in a privacy-preserving manner, we designed a vector encoding method for power prices. For simplicity, we assume that a price is an element selected from an ordered set $P \subset Z$, where Z is the set of integers. The elements in P are sorted in increasing order. We label these elements as $p_1, \ldots, p_{|P|}$, starting from the smallest. For example, if $P = \{11, 12, \ldots, 20\}$, then $p_1 = 11, p_2 = 12, \ldots, p_{10} = 20$. We also define $index_P(p_i)$ where $1 \leq i \leq |P|$. For example, $index_P(13) = 3$. In a situation where decimal fractions are allowed for a price, appropriate scaling and quantization methods can be applied. To apply the FHIPE scheme [50] explained in Section 2.3, we encode a price value $pp_U \in P$ with two $|P|$-dimensional vectors, U^L and U^R, which we call left and right vectors, respectively. We encode U^L, so that its elements with index $< index_P(pp_U)$ are 0, and the other elements are 1. Meanwhile, a one-hot encoding is used for U^R. For example, if $pp_U = 15$, $U^L = (0,0,0,0,1,1,1,1,1,1)$, and $U^R = (0,0,0,0,1,0,0,0,0,0)$. When the prices pp_{U_1} and pp_{U_2} are submitted by two users U_1 and U_2 as (U_1^L, U_1^R) and (U_2^L, U_2^R), respectively, the comparison of pp_{U_1} and pp_{U_2} can be performed by computing an inner product. To be precise, $\langle U_1^L, U_2^R \rangle = 1$ is equivalent to $pp_{U_1} \leq pp_{U_2}$. For example, $pp_{U_1} = 15$ is encoded as

$U_1^L = (0,0,0,0,1,1,1,1,1,1)$ in the above-mentioned example. If $pp_{U_2} = 16$, $\langle U_1^L, U_2^R \rangle = 1$ because $U_2^R = (0,0,0,0,0,1,0,0,0,0)$. If $pp_{U_2} = 14$, $\langle U_1^L, U_2^R \rangle = 0$.

Now, we are ready to design a privacy-preserving matching algorithm. To submit a bid, a user U first encrypts the bid using FHIPE [50] with a secret key sk and random $\alpha, \beta \in \mathbb{Z}_q$ as follows: $E_L(U^L) \leftarrow \mathsf{LeftEncrypt}(\mathsf{sk}, \alpha, U^L)$ and $E_R(U^R) \leftarrow \mathsf{RightEncrypt}(\mathsf{sk}, \beta, U^R)$. If U wants to sell electricity, U submits an encrypted bid as a selling bid. Then, U is added to the seller set **S**, and the pair $(E_L(U^L), E_R(U^R))$ is added to the min-heap. If U's bid is a buying bid, U is added to **B**, and $(E_L(U^L), E_R(U^R))$ is added to the max-heap. The bid values in these heaps must be maintained as ciphertexts. We denote these two encrypted heaps as EH_S and EH_B. We implemented a heap as an array of elements, where each heap element is a structure consisting of an encrypted bid and auxiliary data. When an encrypted bid $(E_L(U^L), E_R(U^R))$ is added to either EH_S or EH_B, the encrypted bids must be compared in the ciphertext domain. This is possible by performing an FHIPE decryption operation. Let $(E_L(U_1^L), E_R(U_1^R))$ and $(E_L(U_2^L), E_R(U_2^R))$ be the encrypted bids of U_1 and U_2, respectively. We have seen in Section 2.3 that $\mathsf{Decrypt}(\mathsf{op}, E_L(U_1^L), E_R(U_2^R))$ gives $\langle U_1^L, U_2^R \rangle$, i.e., $\mathsf{Decrypt}(\mathsf{op}, E_L(U_1^L), E_R(U_2^R)) = 1$ is equivalent to $pp_{U_1} \leq pp_{U_2}$. This operation enables anyone who possesses the public parameters op and ciphertext $(E_L(U_1^L), E_R(U_2^R))$ to compare pp_{U_1} and pp_{U_2}. Note that "Decrypt" does not mean recovering either U_1^L or U_2^R. Therefore, the comparison of $E_L(U_1^L)$ and $E_R(U_2^R)$ is possible without revealing their actual values. As a more intuitive notation, we will use the notation $\mathrm{COMP}(E_L(U_1^L), E_R(U_2^R)) \leftarrow \mathsf{Decrypt}(\mathsf{op}, E_L(U_1^L), E_R(U_2^R))$ hereafter, which returns 1 when $pp_{U_1} \leq pp_{U_2}$. Algorithm 1 is the procedure INSERT for inserting a new element into an encrypted min-heap using COMP. As in a typical complete binary tree implementation, the root node is placed at $EH_S[1]$. The left and right children of $EH_S[i]$ are at $EH_S[2i]$ and $EH_S[2i+1]$, respectively. The parent node of $EH_S[i]$ is at $EH_S[\lfloor i/2 \rfloor]$. It is the same as the legacy heap operation, except that elements are compared over the ciphertext domain. Similarly, the procedure $\mathrm{REMOVEMIN}(EH_S)$ for removing the top element, i.e., the minimum, from the encrypted heap EH_S is defined in a straightforward way. Heap operations for an encrypted max-heap are defined in an analogous way. Algorithm 2 is the procedure for finding a possible match.

4.2. Privacy-Preserving Energy Trading Protocol

In this subsection, we present the proposed privacy-preserving energy trading protocol. The protocol comprises three stages: the setup, bidding and matching, and trading stages.

4.2.1. Setup Stage

Figure 2 shows the setup stage of our protocol, which begins with the DSO performing the Setup algorithm of FHIPE. The DSO generates public parameters op and a secret key sk. In addition, a smart contract for peer matching is created on the blockchain with op. The validity of this smart contract can be verified by any party because op is open to the public. When a user connects a smart meter to the DSO's network, a setup request is generated and forwarded to the DSO. When the DSO receives this request, the pair $(\mathsf{op}, \mathsf{sk})$ is sent to the smart meter. The address of the smart contract is also sent to the smart meter. When the smart meter successfully receives the $(\mathsf{op}, \mathsf{sk})$ pair and the smart contract address, it notifies the user of the setup completion.

Algorithm 1 INSERT procedure for encrypted min-heap EH_S.

Input: $E_L(U^L), E_R(U^R)$, auxiliary data
Output: none
1: $idx \leftarrow$(size of (EH_S))+1 ▷ Insert the new item as the last leaf node.
2: $EH_S[idx].E_L \leftarrow E_L(U^L)$
3: $EH_S[idx].E_R \leftarrow E_R(U^R)$
4: $EH_S[idx].aux \leftarrow$ auxiliary data
5: **while** $idx > 1$ **do** ▷ Perform *upheap* to restore the heap-order.
6: **if** COMP $(EH_S[idx].E_L, EH_S[\lfloor idx/2 \rfloor].E_R) = 1$ **then**
7: **swap** $EH_S[idx]$ **and** $EH_S[\lfloor idx/2 \rfloor]$
8: $idx \leftarrow \lfloor idx/2 \rfloor$
9: **else break**

Algorithm 2 MATCHING procedure to find a possible match between a seller and a buyer.

Input: none
Output: data or $false$
1: **if** COMP $(EH_S[1].E_L, EH_B[1].E_R) = 1$ **then** ▷ matching success
2: $S_m \leftarrow$ REMOVEMIN(EH_S)
3: $B_M \leftarrow$ REMOVEMAX(EH_B)
4: **Return** $(S_m.E_L, S_m.E_R, S_m.aux)$ and $(B_M.E_L, B_M.E_R, B_M.aux)$
5: **else** ▷ matching failure
6: **Return** $false$

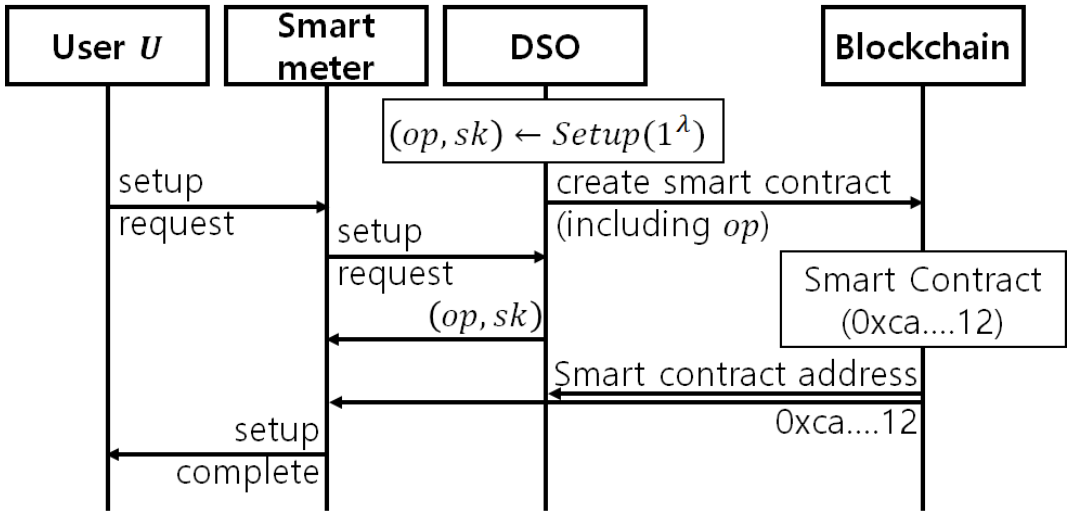

Figure 2. Setup stage of the proposed protocol.

4.2.2. Bidding and Matching Stage

Figure 3 demonstrates the bidding and matching stage. When users want to trade energy, they provide the smart meter with the details of the desired trade, i.e., the intent to buy or sell the electrical power, power amount, and price per unit, which we denote as $intent$, $pa \in Z$, and $pp \in P$, respectively, where P is the set of valid bids. The smart meter randomly generates a one-time identifier OID, which can be considered a pseudonym for privacy. Then, the relation between UID and OID is registered to the DSO for this session, where UID is the permanent ID of the user. pp is encoded into two $|P|$-dimensional vectors U^L and U^R using the encoding method explained in Section 4.1. These two vectors are encrypted using the FHIPE operations as follows: $E_L(U^L) \leftarrow \mathsf{LeftEncrypt}(\mathsf{sk}, \alpha, U^L)$ and

$E_R(U^R) \leftarrow \mathsf{RightEncrypt}(\mathsf{sk}, \beta, U^R)$. We will use a simplified notation $EPP \leftarrow E(\alpha, \beta, pp)$ to cover the entire vector encoding and FHIPE encryption procedures, i.e., $EPP = (E_L(U^L), E_R(U^R))$. In addition, a hash function H is computed to commit to pa, pp, OID, α, and β with a random number r. The smart meter submits the encrypted bid EPP, together with $intent, OID$, and the commitment c to the blockchain by calling the smart contract created by the DSO in the setup phase. Many smart meters submit their encrypted bids similarly. The blockchain maintains EH_S and EH_B to manage these bids. Whenever a new bid EPP is received, either EH_S or EH_B is updated according to Algorithm 1, where auxiliary data contain the $intent, OID$, and c, corresponding to EPP. For simplicity, we denoted this operation as INSERT(EPP) in Figure 3. We denoted the encrypted bid, $intent, OID$, and c submitted by another smart meter as $EPP', intent', OID'$, and c', respectively, to distinguish them from the ones submitted from U's smart meter. These data are also inserted into either EH_S or EH_B according to their $intent'$ value. The submission and insertion of encrypted bids and auxiliary data are performed continuously at the request of smart meters. When both EH_S and EH_B are nonempty and a trigger condition is satisfied, a matching operation is performed on the blockchain using Algorithm 2. With regard to the trigger condition, we may consider various cases. For example, the matching may be performed whenever a new bid is submitted. Matching may also be performed periodically for the waiting bids. If the matching is successful, the encrypted bids and auxiliary data of S_m and B_M are returned by Algorithm 2. Blockchain nodes forward these data to the appropriate smart meters, and each of the two smart meters notifies its user that it was selected for trading. The DSO also obtains the information about the matched users. For simplicity, we will use subscripts S and B to represent a matched seller and buyer, respectively, i.e., the auxiliary data for the matched seller and buyer are $(intent_S, OID_S, c_S)$ and $(intent_B, OID_B, c_B)$, respectively. Their encrypted bids will be written as EPP_S and EPP_B, i.e., $EPP_S = (S_m.E_L, S_m.E_R)$ and $EPP_B = (B_M.E_L, B_M.E_R)$.

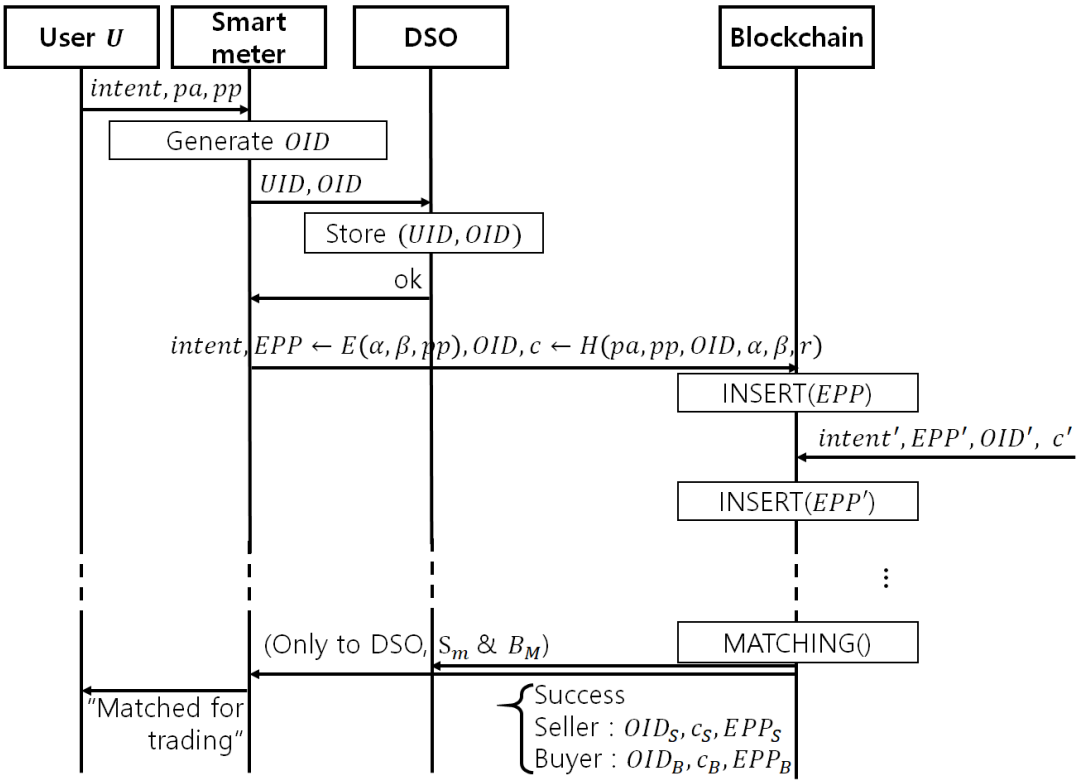

Figure 3. Bidding and matching stage of the proposed protocol.

4.2.3. Trading Stage

Figure 4 demonstrates the trading stage. Let U_S and U_B be the two matched users with OID_S and OID_B, and let SM_S and SM_B be their smart meters, respectively. When a smart meter is informed by the blockchain that its OID has been selected for trading, it sends the DSO all the data for opening the commitment. For example, SM_S transmits $OID_S, pa_S, pp_S, \alpha_S, \beta_S$, and r_S. The DSO verifies that these values are the same as those committed at the bidding and matching stage by comparing the hash value $H(pa_S, pp_S, OID_S, \alpha_S, \beta_S, r_S)$ with the corresponding commitment c_S. The DSO also verifies that the price pp_S has been appropriately encoded in the encrypted bid EPP_S by comparing EPP_S with a reproduced ciphertext $E(\alpha_S, \beta_S, pp_S)$. Similar verifications are repeated for c_B and EPP_B. The DSO then decides a negotiated price PP according to a predefined policy (e.g., the average of pp_S and pp_B.) The DSO forwards all the data received from SM_B to SM_S, and vice versa. PP is also sent to the two smart meters. After verifying the other party's c and EPP, each smart meter verifies that PP conforms to the price-decision policy. The DSO and two smart meters compute the trading amount $PA \leftarrow \min(pa_S, pa_B)$. If all verifications are successful and PA is successfully decided, the smart meters send an "ok" message to the DSO. Then, SM_S feeds energy to the energy storage, and SM_B consumes the energy provided by the energy storage. The energy storage reports the amounts of fed and consumed energy to the DSO, as well as the identifiers SM_S and SM_B of the corresponding smart meters. The DSO combines this information and the stored (UID, OID) relations to identify U_S and U_B and adjusts their account balances. The two users U_S and U_B are notified that their balances were updated.

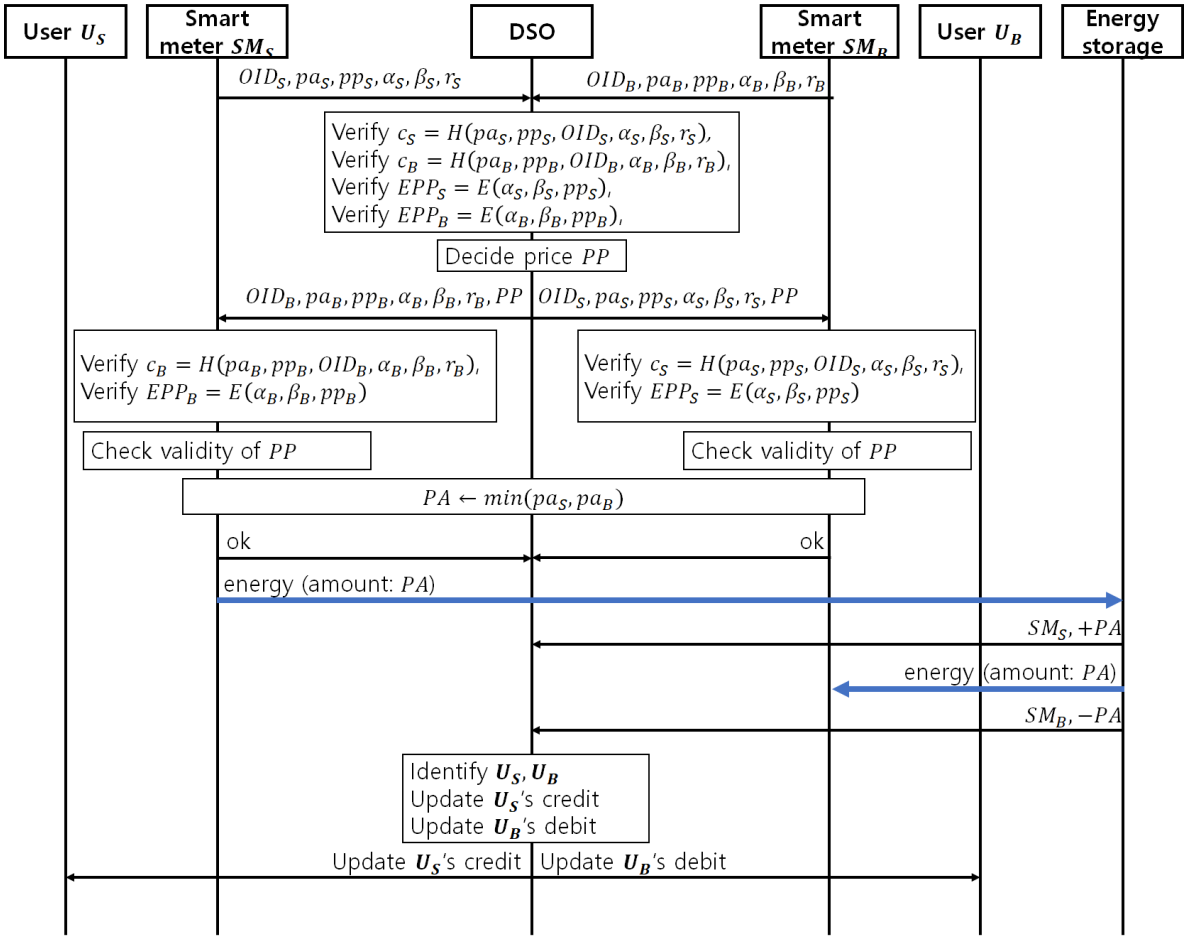

Figure 4. Trading stage of the proposed protocol.

5. Simulation and Performance Analysis

In this section, we verify the feasibility of the proposed system by implementing a prototype. To simulate the system illustrated in Figure 1, we implemented a DSO, private Ethereum blockchain, and smart meter. The DSO was implemented on a desktop server with an Intel Core i7-7700 CPU (3.60 GHz) and 16 GB RAM, and the private Ethereum blockchain was implemented on a desktop server with the same configuration. To simulate a smart meter, we used a Raspberry Pi 2 with an ARM Coretex-A7 quad-core CPU (900 MHz) and 1 GB RAM. The software for the DSO and smart meter was implemented in the C++ programming language, and the blockchain codes were implemented in the Solidity and Go languages. Figure 5 illustrates the internal components of the DSO, blockchain, and smart meter. The DSO contains storage module to store the open parameter op, secret key sk, (UID, OID) pairs, and contract results to calculate the charges associated with users. The blockchain nodes provide storage to maintain two encrypted heaps EH_S and EH_B. The smart meter contains a storage module to store op, sk, permanent UID, smart contract address, and the session information including the one-time identifier OID, bid (pa, pp), and the randomizers (α, β, r) for hiding pp and pa. The smart meter must be equipped with a measurement module. Because the smart meter feeds and receives energy, it should be bi-directional. However, we did not equip the smart meter with the actual power measurement module in our implementation as power measurement was not a primary concern in our experiment. Finally, each party contains an FHIPE module to perform cryptographic operations and a network module to communicate with others. Figure 5 also shows the data flows between the internal components in each stage. The data flows in the setup stage, bidding and matching stage, and the trading stage are shown as blue, red, and green lines, respectively. Most of the data flows depicted in Figure 5 are straightforward according to the protocol description in the previous section. Therefore, we focus on the data exchange involving the FHIPE modules below.

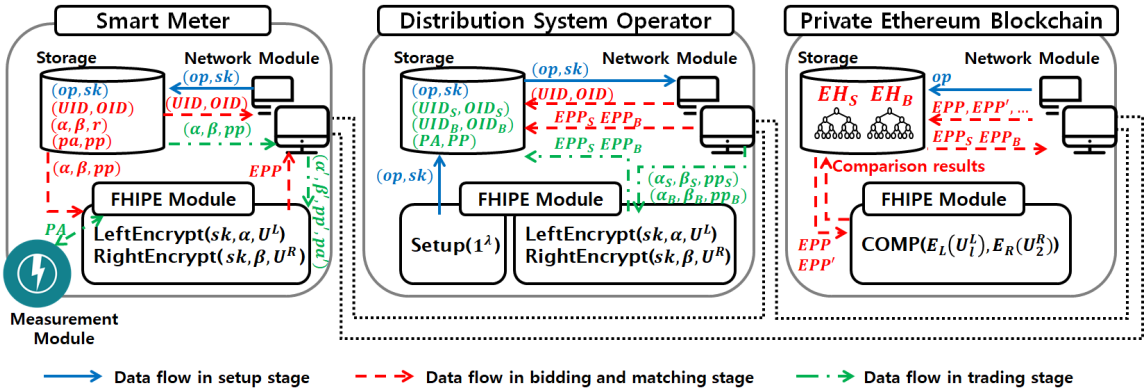

Figure 5. Components of the proposed system. FHIPE = function-hiding inner product encryption.

Because the cryptographic operations performed by the three parties are different, their FHIPE modules contain different algorithms. Specifically, the DSO performs Setup in the setup stage (Figure 2), producing (op, sk). The DSO also performs LeftEncrypt and RightEncrypt for the verification of encrypted bids in the trading stage (Figure 4). Its FHIPE module takes $(\alpha_S, \beta_S, pp_S)$ and $(\alpha_B, \beta_B, pp_B)$ as input and produces EPP_S and EPP_B to be compared with those received from the blockchain in the bidding and matching stage. The FHIPE module in the smart meter performs LeftEncrypt and RightEncrypt operations for generating its encrypted bid EPP in the bidding and matching stage (Figure 3) and verifying the counterpart's bid $(\alpha', \beta', pp', pa')$ in the trading stage (Figure 4). The FHIPE module in the blockchain repeatedly performs COMP, i.e., Decrypt, for the INSERT and MATCHING operations in the bidding and matching stage (Figure 3). The Decrypt operation requires pairing computation e. We used an optimal Ate pairing [53] with a pairing-friendly Barreto-Naehrig (BN) curve [54], as defined by Ethereum.

According to Section 2.3, the Setup, LeftEncrypt, and RightEncrypt operations performed by the DSO and smart meter require the implementation of group operations on $\mathbb{G}_1$ and $\mathbb{G}_2$. To be precise, these operations are point operations on a pairing-friendly BN curve because $\mathbb{G}_1$ and $\mathbb{G}_2$ are BN curve groups that enable pairing operations on the blockchain. To implement the group operations, we also must implement the underlying finite field operations and big number operations. We used the pairing-based cryptography library (MCL) [55], for elliptic curve operations, the GNU Multiple Precision Arithmetic Library (GMP) [56], for big number arithmetic operations, and the library for doing number theory (NTL) [57], for operations over a finite field, vector, and matrix. The MCL library supports optimizations for BN curves in two layers. First, it supports the generalized lazy reduction technique and the new squaring formula proposed in Reference [58] for the underlying finite field. Second, in a higher layer, it supports an efficient scalar multiplication method using the skew Frobenius map on BN curves [59]. For the smart meter, we implemented the FHIPE module, applying these optimization methods. For the DSO, we adopted the FHIPE module in Reference [60], where the above optimization methods, as well as parallel processing, were applied. For the Decrypt operation performed by the blockchain, we used the pre-compiled contract provided by Ethereum.

Now, we present our experimental results. Table 1 summarizes the performance of FHIPE operations by each party for $|P|$ = 10, 20, 30, 40, and 50, where $|P|$ is the number of possible choices for power price per unit, i.e., the dimension of the left and right vectors. The values in the table are the averages of 1000 iterations for each parameter combination. We observed that the smart meter required significantly more time than the DSO for performing the same operations, i.e., LeftEncrypt and RightEncrypt. For example, when $|P| = 10$, LeftEncrypt in the smart meter required 62.040 milliseconds, but the same operation in DSO was performed in 0.120 ms, which is 0.19% of the time consumed by the smart meter. RightEncrypt consumed 351.538 ms in the smart meter and 0.566 ms in the DSO. The DSO required only 0.16% of the time required by the smart meter because the smart meter resource is very constrained compared to the DSO, and an additional optimization method, i.e., parallel processing, was applied to the DSO. However, it must be noted that all operations were completed in less than 2 s, even in the smart meter.

Table 1. Measured times for performing FHIPE operations in smart meter, DSO, and blockchain (milliseconds).

Range of Price	Smart Meter			DSO		Blockchain
	LeftEncrypt	RightEncrypt	Setup	LeftEncrypt	RightEncrypt	COMP
$\|P\| = 10$	62.040	351.538	0.434	0.120	0.566	10.679
$\|P\| = 20$	115.252	654.209	0.952	0.122	0.843	19.932
$\|P\| = 30$	176.429	999.862	2.076	0.135	1.239	29.167
$\|P\| = 40$	224.141	1271.670	4.030	0.146	1.524	38.209
$\|P\| = 50$	282.103	1595.647	7.026	0.159	1.940	47.053

Next, we analyzed the performance of the proposed protocol. First, we evaluated the bidding and matching stage by measuring the transaction throughput, where a transaction covers the following portion of Figure 3: the smart meter's encrypting a bid into EPP, computing the commitment c, and sending $intent, EPP, OID$, and c to the blockchain, and a blockchain node's mining a block containing the execution of INSERT and MATCHING procedure, and sending the result back to the smart meter. In the experiment, we considered two setups. For the first setup, we measured the throughput with a single smart meter. The second and third columns in Table 2 present the experimental result in two units: transactions per second (tps) and kilobits per second (kbps). The values are the averages for 1000 bids, where the bid values were randomly selected from P, and the intent to either buy or sell was also randomly selected for each bid. The results indicate that an entire transaction is completed in roughly a second when $|P| = 10$ and in a little more than 4 s even when $|P| = 50$.

Table 2. Transaction throughput of the proposed protocol.

Range of Price	Throughput with a Single Smart Meter		Throughput with Multiple Smart Meters		Throughput of Non-Encryption Protocol with Multiple Smart Meters	
	In tps	In kbps	In tps	In kbps	In tps	In kbps
$\|P\| = 10$	0.981	21.248	2.010	41.656	58.882	446.32
$\|P\| = 20$	0.552	20.950	0.990	36.259	34.517	438.49
$\|P\| = 30$	0.378	20.561	0.606	31.948	25.025	446.34
$\|P\| = 40$	0.289	20.496	0.423	29.100	17.561	403.28
$\|P\| = 50$	0.234	20.186	0.330	27.852	13.712	385.26

The second setup is for multiple smart meters. Recall that in the first setup, the measured time covered the operations for both the smart meter and the blockchain. However, in a more realistic situation, the blockchain does not have to wait until a smart meter completes its task. Instead, it is reasonable to assume that many smart meters generate encrypted bids in parallel, and the blockchain continues its transactions without stalling. Therefore, to simulate this situation, we pre-generated 1000 bids and fed them to the blockchain. According to the experimental results presented in the fourth and fifth columns of the table, when $|P| = 10$, the throughput was doubled compared to the single smart meter case. When $|P| = 50$, the throughput increased by 41%. For reference, we also presented the measured throughputs when the bid protection was not applied, i.e., the bids were not encrypted. The last two columns of Table 2 show that the transactions were performed up to $13.712/0.330 \approx 41.55$ times faster when the bidding and matching stage was implemented without encryption.

We also analyzed the Ethereum gas consumption in the bidding and matching stage of the proposed protocol. In the protocol, the two dominant operations that consume gas are heap node creation and COMP operation. First, when an encrypted bid is inserted into an encrypted heap according to Algorithm 1, a new leaf node containing $(EPP, intent, OID, c)$ is created, which requires non-negligible memory, thereby consuming gas. Second, according to the specification of go-ethereum [61], which is an EVM implemented with the Go language, a combined pairing operation involving an n-dimensional vector consumes $80{,}000n + 100{,}000$ gas [62]. In our case, a COMP operation requires a combined pairing operation with $n = |P|$ and some additional operations. The COMP operation is used to manage two encrypted heaps, EH_S and EH_B, i.e., it is repeatedly used for the INSERT and MATCHING procedures, as shown in Algorithm 1 and Algorithm 2. Although we did not demonstrate an explicit algorithm, the REMOVEMIN procedure also repeatedly calls COMP. Table 3 details the gas consumption of these two dominating operations. We measured the consumed gas for 100 iterations, and the values presented in the table are the averages. It can be inferred from the experimental result that the gas consumption of heap node creation increased by a similar amount each time $|P|$ was increased by 10. For example, the difference in gas consumption between $|P| = 10$ and $|P| = 20$ was 1,334,272, and the difference between $|P| = 20$ and $|P| = 30$ was 1,334,236. We also observe that the gas consumption of COMP was slightly greater than $80{,}000 \times |P| + 100{,}000$.

Table 3. Amount of gas consumed to perform a heap node creation and COMP operation.

Consumed Gas / Range of Price	Heap Node Creation	COMP
$\|P\| = 10$	1,626,775	941,188
$\|P\| = 20$	2,961,047	1,594,308
$\|P\| = 30$	4,295,283	2,508,970
$\|P\| = 40$	5,629,511	3,415,960
$\|P\| = 50$	6,964,833	4,175,020

Regarding the performance of the trading stage, the dominant operations for this stage are LeftEncrypt and RightEncrypt operations of the DSO and two smart meters, because other operations,

such as hash computation, are negligible. As the operations of the two smart meters are independently performed in parallel, the time required for this stage is roughly 2 × (DSO's left and right encryption time) + (smart meter's left and right encryption time). It is roughly 415 ms for $|P| = 10$ and 1.88 s for $|P| = 50$ according to Table 1.

For an objective evaluation of the performance of the proposed system, we compare it with the performance of previous blockchain-based privacy-perserving applications. Because these applications were not for P2P energy trading, direct comparison with the proposed system is not possible. However, they can be the reference for the performance of our system. First, we examine Zerocash [22] that ensures user anonymity for cryptocurrency transfer. The main cryptographic primitive of Zerocash is zero-knowledge Succinct Non-interactive Arguments of Knowledge (zk-SNARKs), which are composed of KeyGen, Prove, and Verify algorithms. In Reference [22], the following six functions for Zerocash were defined by using these algorithms; Setup, CreateAddress, Mint, Pour, VerifyTransaction, and Receive. Among them, the Pour function actually transfers cryptocurrency. According to the experimental results on an Intel i7-4770 @ 3.40 GHZ CPU and 16 GB RAM, the Prove algorithm needed 2 min 2 s, while the Verify algorithm needed only 5.4 ms [22]. Because the dominant operation of the Pour function is Prove, a Pour operation needed 2 min 2 s. In contrast, the dominant blockchain operation in the proposed system is COMP, which is based on the similar operation to the underlying operation of Verify of Zerocash. We already verified in Tables 1 and 2 that COMP is completed in tens of milliseconds and an entire transaction is completed in a few seconds.

Next, we examine the performance of Hawk [24]. Hawk is a more general framework for building privacy-preserving smart contracts. As in Zerocash, the KeyGen, Prove and Verify algorithms were defined for zk-SNARKs, and they were used to design Pour, Freeze, Compute, and Finalize operations. Among the applications of Hawk presented in Reference [24], we may refer to the auction application. This application is composed of user-side and manager-side operations. According to the experimental results on Amazon EC2 r3.8xlarge virtual machines in a 27 GB, 4-core environment, the manager-side Finalize operation for selecting a winner required 469.7 s for 100 participants. According to Table 2, the proposed system requres only $100/2.010 \approx 49.75$ s for 100 transactions when $|P| = 10$ and $100/0.330 \approx 303.03$ s when $|P| = 50$.

6. Security Analysis

In this section, we examine whether the proposed system satisfies the security goals defined in Section 3.2.

- **User Privacy**: Users are identified on the blockchain with the one-time pseudonym OID of their smart meter for each trading session. Because OID is an ephemeral identifier, the other peers connected to the blockchain do not learn the user's permanent identifier UID. In addition, the activities of the same user in two different sessions cannot be linked because the OID is refreshed every session. The UID is revealed to the DSO through the initiation message from the smart meter to the DSO in Figure 3, but this is not a violation of our security goal.
- **DSO Privacy**: What the blockchain nodes see in the bidding and matching stage are the tuples $(intent, EPP, OID, c)$ for each bid and the matching results. Although EPP and c contain the pp and pa for the bid, they are protected by the FHIPE and cryptographic hash function. From the one-way property of the cryptographic hash function, it is evident that the pa for the bid is protected. The proof that pp is well protected will be presented in the Appendix A. In the trading stage, pa and pp values are opened to the two smart meters, which is necessary for energy trading. However, these values are protected from the other parties because all data communication lines are protected by an end-to-end encryption mechanism according to the system model in Section 3.1.
- **Verifiability of transactions** and **integrity of bids**: Because the blockchain is honest-but-curious, it honestly performs heap updates (Algorithm 1) and peer matching (Algorithm 2). Therefore, the seller with the minimum bid and the buyer with the maximum bid are always matched with each

other. Whether the operations are being performed correctly is verifiable by any user connected to a blockchain node, although the actual contents of the bids are protected by FHIPE. With regard to the integrity of a bid, the blockchain guarantees integrity by nature. Therefore, no one can modify the data on the blockchain, such as $intent$, EPP, OID, and c, without properly calling smart contracts. According to Figure 3, c_B on the blockchain is sent to SM_S. Therefore, if the DSO attempts to modify either pa_B or pp_B in Figure 4, SM_S will notice this modification through the verification of c_B. Additionally, by comparing EPP_B with $E(\alpha_B, \beta_B, pp_B)$, SM_S can verify that pp_B had been properly encoded in the EPP_B that was used for comparison and matching. SM_B can verify the integrity of pa_S and pp_S in the same way.

- **Nonrepudiation of bids**: Because smart meters are tamper-proof, the users cannot revoke their bids by manipulating the smart meter. Furthermore, once the encrypted bid EPP and commitment c are uploaded on the blockchain, their integrity is protected by the blockchain. Because c is bound to pa and pp using a cryptographic hash function, neither pa nor pp can be denied.

7. Discussion

In this paper, we proposed a P2P energy trading system on a blockchain where peer matching is performed on the encrypted bids by a functional encryption-based smart contract. We also verified the feasibility of the proposed system by implementing a prototype composed of smart meters, a DSO, and private Ethereum blockchain. However, the system could be improved in several ways. First, we only considered the case where new encrypted bids are inserted into the heap. However, it would be desirable if there were a mechanism to remove a bid when it expires. Second, the dimension of the vectors encoding bid values is proportional to the range of possible bid values. If we can use a more compact encoding, e.g., a binary encoding, we can significantly reduce the time and gas consumption for bid generation and peer-matching. Finally, the smart meter performance can be improved by adopting more implementation optimization. We shall address these issues in future research.

The proposed method provides the means to hide the electricity price and amount contained in a bid, but it does not tell a prosumer what price to bid. Many relevant studies have been conducted to improve the energy management and optimize the costs. Various techniques, such as fuzzy logic [63], cooperative game theory [64], genetic algorithms [65], and deep recurrent neural networks [66], have been proposed. Specifically, a reinforcement learning-based strategy (Fuzzy Q-learning) was recently proposed to improve the decision-making process of P2P power trading [67]. These strategies may be combined with our privacy-enhancing method.

Furthermore, the proposed system may be combined with other privacy-preserving techniques for smart grids. In recent years, extensive research on smart meter data obfuscation has been conducted to prevent the leakage of smart meter readings [37–39]. However, these methods are adding noise to the measurements; thus, they affect the billing and control functionalities [68]. Therefore, many studies have proposed the use of batteries to mask the energy-consumption statistics [68–70]. The advantage of using batteries is that they not only improve privacy but also reduce energy costs when charging and discharging are cleverly controlled. For this purpose, multiobjective optimization methods have been proposed to simultaneously minimize privacy leakage and energy cost [69,70]. However, these proposals assume that energy prices are set up by the utility company, and the prosumer's active pricing mechanism with P2P trading has not been considered. Therefore, it would be very interesting if we combined the proposed system with these battery-enabled data protection methods. This may ensure privacy in two aspects: protection of power-consumption and P2P transaction data.

Author Contributions: Y.-B.S. designed the protocol and implemented the blockchain module of the prototype system. J.-H.I. designed the strategy for applying FHIPE to the proposed system. H.-Y.K. planned the experiments and validated the experimental results. S.-Y.J. implemented the smart meter module of the proposed system. M.-K.L. proposed the main idea for this research and organized the entire process. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by Korea Electric Power Corporation (Grant number: R18XA01) and in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2020R1A2C2013089).

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

Abbreviations

The following abbreviations are used in this manuscript:

BN	Barreto-Naehrig
DSO	Distribution System Operator
EVM	Ethereum Virtual Machine
FE	Functional Encryption
FHIPE	Function-Hiding Inner Product Encryption
LSE	Load Serving Entity
NIALM	Non-Intrusive Appliance Load Monitoring
OID	One-time IDentifier
PPT	Probabilistic Polynomial Time
P2P	peer-to-peer
TSO	Transmission System Operator
UID	User's permanent IDentifier
zk-SNARKs	zero-knowledge Succinct Non-interactive Arguments of Knowledge
tps	transactions per second
kbps	kilobits per second

Appendix A. Security Proof

In this section, we prove that pp is well protected by FHIPE. We first review the security properties of the FHIPE scheme Π_{IPE} in Reference [50]. Next, we prove the security of the proposed protocol using reduction from Π_{IPE}.

Appendix A.1. Review of the Security of the FHIPE scheme Π_{IPE}

Previous works on inner product encryption, including Π_{IPE} [50] we used considered an indistinguishability notion of security [50]. We review the security notion of Π_{IPE} defined in Reference [50]. Specifically, an experiment between a challenger and an adversary $\mathcal{A}$ that can make the left and right encryption oracle queries is defined as follows.

Definition A1 (Experiment $\mathsf{Expt}_b^{\mathsf{IPE-IND}}$ [50])**.** *Let* $b \in \{0,1\}$*. The challenger computes* $(\mathsf{op}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$*, gives* op *to the adversary* $\mathcal{A}$*, and then responds to each oracle query type made by* $\mathcal{A}$ *in the following manner.*

- **Left encryption oracle.** *On input a pair of vectors* $\mathbf{x}_0, \mathbf{x}_1 \in \mathbb{Z}_q^n \setminus \{0\}$*, the challenger computes and returns* $E^L(\mathbf{x}_b) \leftarrow \mathsf{LeftEncrypt}(\mathsf{sk}, \alpha, \mathbf{x}_b)$ *using a random element* $\alpha \in \mathbb{Z}_q$*.*
- **Right encryption oracle.** *On input a pair of vectors* $\mathbf{y}_0, \mathbf{y}_1 \in \mathbb{Z}_q^n \setminus \{0\}$*, the challenger computes and returns* $E^R(\mathbf{y}_b) \leftarrow \mathsf{RightEncrypt}(\mathsf{sk}, \beta, \mathbf{y}_b)$ *using a random element* $\beta \in \mathbb{Z}_q$*.*

Eventually, $\mathcal{A}$ *outputs a bit* b'*, which is also the output of the experiment, denoted by* $\mathsf{Expt}_b^{\mathsf{IPE-IND}}(\mathcal{A})$*.*

Then, the security of an FHIPE scheme is defined using an indistinguishability notion as follows:

Definition A2 (Admissibility of $\mathcal{A}$ [50])**.** *For an adversary* $\mathcal{A}$*, let* Q_L *and* Q_R *be the total number of left and right encryption oracle queries made by* $\mathcal{A}$*, respectively. For* $b \in \{0,1\}$*, let* $\mathbf{x}_b^{(1)}, \ldots, \mathbf{x}_b^{(Q_L)} \in \mathbb{Z}_q^n \setminus \{0\}$ *and* $\mathbf{y}_b^{(1)}, \ldots, \mathbf{y}_b^{(Q_R)} \in \mathbb{Z}_q^n \setminus \{0\}$ *be the corresponding vectors that* $\mathcal{A}$ *submits to the left and right encryption oracles, respectively. We say that* $\mathcal{A}$ *is admissible if for all* $i \in \{1, \ldots, Q_L\}$ *and* $j \in \{1, \ldots, Q_R\}$*, we have that:*

$$\langle \mathbf{x}_0^{(i)}, \mathbf{y}_0^{(j)} \rangle = \langle \mathbf{x}_1^{(i)}, \mathbf{y}_1^{(j)} \rangle.$$

Definition A3 (IND-Security for IPE [50]). *We define an inner product encryption scheme denoted as* $\Pi_{IPE} = (\mathsf{Setup}, \mathsf{LeftEncrypt}, \mathsf{RightEncrypt}, \mathsf{Decrypt})$ *as fully-secure if for all efficient and admissible adversaries* $\mathcal{A}$*:*

$$\left|\Pr[\mathsf{Expt}_0^{\mathsf{IPE-IND}}(\mathcal{A}) = 1] - \Pr[\mathsf{Expt}_1^{\mathsf{IPE-IND}}(\mathcal{A}) = 1]\right| = \mathsf{negl}(\lambda),$$

where $\mathsf{negl}(\lambda)$ *denotes a negligible function in* λ*.*

Theorem A1 ([50]). *The inner product encryption scheme* Π_{IPE} *is* IND*-secure in the generic group model.*

Remark A1. *The original statement in Theorem 7 in Reference [50] is that the inner product encryption scheme* Π_{IPE} *is* SIM*-secure in the generic group model. It was also remarked in Remark 5 in Reference [50] that a* SIM*-secure IPE scheme is also* IND*-secure. We merged these two statements into Theorem A1. See Reference [50] for more information on the* SIM*-security and a generic group model.*

Appendix A.2. Proof of the Security of the Proposed System

We denote the proposed privacy-preserving P2P energy trading protocol by Π_{PPET}. Our threat model in Section 3.2 assumed that a blockchain is honest-but-curious. That is, the blockchain nodes might attempt to extract useful information regarding the power price pp of a user. Regarding the security against the honest-but-curious blockchain, we define an indistinguishability notion of security for Π_{PPET}. Then, we prove the IND-security of Π_{PPET} using that of Π_{IPE}. Our proof is a modification of the security proof of two-input functional encryption in the full version of Reference [50].

First, we define the following experiment between a challenger and an adversary $\mathcal{A}^*$ that can make two oracle queries, LeftEncrypt and RightEncrypt.

Definition A4 (Experiment $\mathsf{Expt}_b^{\mathsf{PPET-IND}}$). *Let* $b \in \{0,1\}$*. The challenger computes* $(\mathsf{op}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$ *using* Π_{IPE}*, gives* op *to the adversary* $\mathcal{A}^*$*, and then responds to each oracle query type made by* $\mathcal{A}^*$ *in the following manner.*

- **LeftEncrypt oracle.** *On input a pair of two messages* $x_0, x_1 \in \{z, \ldots, z + (n-1)\}$ *for a positive integer* z*, the challenger encodes* x_0, x_1 *to* $U_0^L, U_1^L \in \mathbb{Z}_q^n \setminus \{0\}$ *according to the description of Section 4.1, computes and returns* $E_L(U_b^L) \leftarrow \mathsf{LeftEncrypt}(\mathsf{sk}, \alpha, U_b^L)$ *using a random element* $\alpha \in \mathbb{Z}_q$*.*
- **RightEncrypt oracle.** *On input a pair of two messages* $y_0, y_1 \in \{z, \ldots, z + (n-1)\}$ *for a positive integer* z*, the challenger encodes* y_0, y_1 *to* $U_0^R, U_1^R \in \mathbb{Z}_q^n \setminus \{0\}$ *according to the description of Section 4.1, computes and returns* $E_R(U_b^R) \leftarrow \mathsf{RightEncrypt}(\mathsf{sk}, \beta, U_b^R)$ *using a random element* $\beta \in \mathbb{Z}_q$*.*

Eventually, $\mathcal{A}^*$ *outputs a bit* b'*, which is also the output of the experiment, denoted by* $\mathsf{Expt}_b^{\mathsf{PPET-IND}}(\mathcal{A}^*)$*.*

Definition A5 (Admissibility of $\mathcal{A}^*$). *For an adversary* $\mathcal{A}^*$*, let* Q_L *and* Q_R *be the total number of LeftEncrypt and RightEncrypt oracle queries made by* $\mathcal{A}^*$*, respectively. For* $b \in \{0,1\}$ *and a positive integer* z*, let* $x_b^{(1)}, \ldots, x_b^{(Q_L)} \in \{z, \ldots, z + (n-1)\}$ *and* $y_b^{(1)}, \ldots, y_b^{(Q_R)} \in \{z, \ldots, z + (n-1)\}$ *be the corresponding messages that* $\mathcal{A}^*$ *submits to the LeftEncrypt and RightEncrypt oracles, respectively. We say that* $\mathcal{A}^*$ *is admissible if for all* $i \in \{1, \ldots, Q_L\}$ *and* $j \in \{1, \ldots, Q_R\}$*, we have that:*

$$f(x_0^{(i)}, y_0^{(j)}) = f(x_1^{(i)}, y_1^{(j)}),$$

where f *is a function that computes the inner product of the two vectors encoded from the two input integers, i.e.,* $f(x_b^{(i)}, y_b^{(i)}) = \langle (U_b^L)^{(i)}, (U_b^R)^{(i)} \rangle$*.*

Definition A6 (IND-Security for Π_{PPET}). *We define a privacy-preserving P2P energy trading system* Π_{PPET} *as fully-secure if for all efficient and admissible adversaries* $\mathcal{A}^*$*:*

$$\left|\Pr[\mathsf{Expt}_0^{\mathsf{PPET-IND}}(\mathcal{A}^*) = 1] - \Pr[\mathsf{Expt}_1^{\mathsf{PPET-IND}}(\mathcal{A}^*) = 1]\right| = \mathsf{negl}(\lambda).$$

Theorem A2. *If Π_{IPE} is* IND*-secure (according to Definition A3) in the generic group model, then Π_{PPET} constructed with Π_{IPE} is* IND*-secure (according to Definition A6) in the generic group model.*

Proof of Theorem A2. To perform a reduction proof of the security for Π_{PPET}, assume that there exists an efficient and admissible adversary $\mathcal{A}^*$. We will show that $\mathcal{A}^*$ can be used as a subroutine for the adversary $\mathcal{A}$. That is, we design $\mathcal{A}$ that simulates the LeftEncrypt and RightEncrypt oracles of $\mathsf{Expt}_b^{\mathsf{PPET-IND}}$ by forwarding $\mathcal{A}^*$'s corresponding queries to the left and right encryption oracles of $\mathsf{Expt}_b^{\mathsf{IPE-IND}}$, respectively. Our construction of $\mathcal{A}$ is shown in Algorithm A1. It is straightforward to see that if $\mathcal{A}^*$ is an admissible, polynomial time algorithm, Algorithm A1 is so, too. In addition, $\left|\Pr[\mathsf{Expt}_0^{\mathsf{IPE-IND}}(\mathcal{A})=1]-\Pr[\mathsf{Expt}_1^{\mathsf{IPE-IND}}(\mathcal{A})=1]\right| = \left|\Pr[\mathsf{Expt}_0^{\mathsf{PPET-IND}}(\mathcal{A}^*)=1]-\Pr[\mathsf{Expt}_1^{\mathsf{PPET-IND}}(\mathcal{A}^*)=1]\right|$. However, the construction of $\mathcal{A}$ contradicts Theorem A1, which states that an efficient and admissible $\mathcal{A}$ with non-negligible advantage does not exist. This completes the proof. □

Algorithm A1 Construction of $\mathcal{A}$ using $\mathcal{A}^*$.

Input: public parameters op.
Output: a bit b'.

Give op to $\mathcal{A}^*$.
while true **do**
  if $\mathcal{A}^*$ returns b' **then**
    Return b'.
  Wait until $\mathcal{A}^*$ submits a query $\mathcal{Q}$.
  if $\mathcal{Q}=(x_0,x_1)$ is a LeftEncrypt oracle query **then**
    Encode $x_0,x_1\in\{z,\ldots,z+(n-1)\}$ to $U_0^L,U_1^L\in\mathbb{Z}_q^n\setminus\{0\}$.
  ▷ z is a positive integer and U_b^L is a vector encoded according to the description of Section 4.1.
    Submit (U_0^L,U_1^L) to the left encryption oracle in $\mathsf{Expt}_b^{\mathsf{IPE-IND}}$ and receive $E^L(U_b^L)$.
    Provide $E^L(U_b^L)$ to $\mathcal{A}^*$.
  else ▷ $\mathcal{Q}=(y_0,y_1)$ is a RightEncrypt oracle query.
    Encode $y_0,y_1\in\{z,\ldots,z+(n-1)\}$ to $U_0^R,U_1^R\in\mathbb{Z}_q^n\setminus\{0\}$.
  ▷ z is a positive integer and U_b^R is a vector encoded according to the description of Section 4.1.
    Submit (U_0^R,U_1^R) to the left encryption oracle in $\mathsf{Expt}_b^{\mathsf{IPE-IND}}$ and receive $E^R(U_b^R)$.
    Provide $E^R(U_b^R)$ to $\mathcal{A}^*$.

References

1. Abdella, S.; Shuaib, K. Peer to Peer Distributed Energy Trading in Smart Grids: A Survey. *Energies* **2018**, *11*, 1560. [CrossRef]
2. Wang, S.; Taha, A.F.; Wang, J.; Kvaternik, K.; Hahn, A. Energy Crowdsourcing and Peer-to-Peer Energy Trading in Blockchain-Enabled Smart Grids. *IEEE Trans. Syst. Man Cybern. B Cybern.* **2019**, *49*, 1612–1623. [CrossRef]
3. Fell, M.J.; Schneiders, A.; Shipworth, D. Consumer Demand for Blockchain-Enabled Peer-to-Peer Electricity Trading in the United Kingdom: An Online Survey Experiment. *Energies* **2019**, *12*, 3913. [CrossRef]
4. Shipworth, D. Peer to Peer Distributed Energy Trading Using Blockchains. Available online: http://www.ieadsm.org/wp/files/IEA-DSM-Spotlight-Issue67-December20171.pdf (accessed on 30 January 2020).
5. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. Available online: https://bitcoin.org/bitcoin.pdf (accessed on 30 January 2020).
6. Shipworth, D. An Explorative Study on the Implications of Prosumer-Consumer Communities on the Value Creation in the future Electricity Network. Available online: https://doc.rero.ch/record/277573/files/GstreinM.pdf (accessed on 30 January 2020).

7. Aitzhan, N.Z.; Svetinovic, D. Security and Privacy in Decentralized Energy Trading Through Multi-Signatures, Blockchain and Anonymous Messaging Streams. *IEEE Trans. Dependable Secur. Comput.* **2018**, *15*. [CrossRef]
8. Son, Y.B. Data-Protected Blockchain Using Inner Product Functional Encryption. Master's Thesis, Inha University, Incheon, Korea, 2020. (In Korean)
9. Yuan, Y.; Wang, F.Y. Blockchain and Cryptocurrencies: Model, Techniques, and Applications. *IEEE Trans. Syst. Man Cybern. B Cybern.* **2018**, *48*. [CrossRef]
10. Ethereum. Available online: https://ethereum.org/ (accessed on 23 January 2020).
11. Ethereum White Paper. Available online: https://github.com/ethereum/wiki/wiki/white-paper (accessed on 23 January 2020).
12. Dannen, C. *Introducing Ethereum and Solidity: Foundations of Cryptocurrency and Blockchain Programming for Beginners*, 1st ed.; Apress: New York, NY, USA, 2017.
13. Luu, L.; Chu, D.; Olickel, H.; Saxena, P.; Hobor, A. Making smart contracts smarter. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications (CCS 2016), Vienna, Austria, 24–28 October 2016; pp. 254–269.
14. Solidity Documentation. Available online: https://solidity.readthedocs.io/en/latest (accessed on 23 January 2020).
15. Wood, G. Ethereum: A Secure Decentralised Generalised Transaction Ledger Byzantium Version. Available online: https://ethereum.github.io/yellowpaper/paper.pdf (accessed on 30 January 2020).
16. Feng, Q.; He, D.; Zeadally, S.; Khan, M.K.; Kumar, N. A survey on privacy protection in blockchain system. *J. Netw. Comput. Appl.* **2019**, *126*. [CrossRef]
17. Zhang, R.; Xue, R.; Liu, L. Security and Privacy on Blockchain. *ACM Comput. Surv.* **2019**, *52*, 1–34. [CrossRef]
18. Bonneau, J.; Narayanan, A.; Miller, A.; Clark, J.; Kroll, J.A.; Felten, E.W. Mixcoin: Anonymity for bitcoin with accountable mixes. In Proceedings of the 18th International Conference Financial Cryptography and Data Security (FC2014), Christ Church, Barbados, 3–7 March 2014; pp. 486–504.
19. Ruffing, T.; Moreno-Sanchez, P.; Kate, A. CoinShuffle: Practical decentralized coin mixing for bitcoin. In Proceedings of the 19th European symposium on Research in Computer Security (ESORICS 2014), Wroclaw, Poland, 7–11 September 2014; pp. 345–364.
20. Ben-Sasson, E.; Chiesa, A.; Tromer, E.; Virza, M. Succinct non-interactive zero knowledge for a Von Neumann Architecture. In Proceedings of the 23rd USENIX Security Symposium 2014, San Diego, CA, USA, 20–22 August 2014; pp. 781–796.
21. Bünz, B.; Bootle, J.; Boneh, D.; Poelstra, A.; Wuille, P.; Maxwell, G. Bulletproofs: Short proofs for confidential transactions and more. In Proceedings of the 39th IEEE Symposium on Security and Privacy (SP 2018), San Francisco, CA, USA, 20–24 May 2018; pp. 315–334.
22. Ben-Sasson, E.; Chiesa, A.; Garman, C.; Green, M.; Miers, I.; Tromer, E.; Virza, M. Zerocash: Decentralized anonymous payments from bitcoin. In Proceedings of the 35th IEEE Symposium on Security and Privacy (SP 2014), Berkeley, CA, USA, 18–21 May 2014; pp. 459–474.
23. Bünz, B.; Agrawal, S.; Zamani, M.; Boneh, D. Zether: Towards Privacy in a Smart Contract World. Available online: https://crypto.stanford.edu/~buenz/papers/zether.pdf (accessed on 28 January 2020).
24. Kosba, A.E.; Miller, A.; Shi, E.; Wen, Z.; Papamanthou, C. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In Proceedings of the 37th IEEE Symposium on Security and Privacy (SP 2016), San Jose, CA, USA, 22–26 May 2016; pp. 839–858.
25. Megha, S.; Lamptey, J.; Salem, H.; Mazzara, M. A Survey of of Blockchain-Based Solutions for Energy Industry. Available online: https://arxiv.org/pdf/1911.10509.pdf (accessed on 30 January 2020).
26. Mollah, M.B.; Zhao, J.; Niyato, D.; Lam, K.Y.; Zhang, X.; Ghias, A.M.Y.M.; Koh, L.H.; Yang, L. Blockchain for Future Smart Grid: A Comprehensive Survey. Available online: https://arxiv.org/pdf/1911.03298.pdf (accessed on 30 January 2020).
27. Power Ledger. Available online: https://www.powerledger.io (accessed on 28 January 2020).
28. Lo3. Available online: https://lo3energy.com/ (accessed on 28 January 2020).
29. Brooklyn Microgrid. Available online: https://www.brooklyn.energy/ (accessed on 28 January 2020).
30. Slock.it. Available online: https://slock.it/(accessed on 28 January 2020).
31. SolarCoin. Available online: https://solarcoin.org/(accessed on 28 January 2020).

32. Su, Z.; Wang, Y.; Xu, Q.; Fei, M.; Tian, Y.C.; Zhang, N. A Secure Charging Scheme for Electric Vehicles With Smart Communities in Energy Blockchain. *IEEE Internet Things J.* **2019**, *6*, 4601–4613. [CrossRef]
33. Hahn, A.; Singh, R.; Liu, C.C.; Chen, S. Smart contract-based campus demonstration of decentralized transactive energy auctions. In Proceedings of the IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT 2017), Washington, DC, USA, 23–26 April 2017; pp. 1–5.
34. Li, Z.; Kang, J.; Yu, R.; Ye, D.; Deng, Q.; Zhang, Y. Consortium Blockchain for Secure Energy Trading in Industrial Internet of Things. *IEEE Trans. Ind. Inform.* **2017**, *14*, 3690–3700. [CrossRef]
35. Kang, J.; Yu, R.; Huang, X.; Maharjan, S.; Zhang, Y.; Hossain, E. Enabling Localized Peer-to-Peer Electricity Trading Among Plug-in Hybrid Electric Vehicles Using Consortium Blockchains. *IEEE Trans. Ind. Inform.* **2017**, *13*, 3154–3164. [CrossRef]
36. Gai, K.; Wu, Y.; Zhu, L.; Qiu, M.; Shen, M. Privacy-Preserving Energy Trading Using Consortium Blockchain in Smart Grid. *IEEE Trans. Ind. Inform.* **2019**, *15*, 3548–3558. [CrossRef]
37. Bohli, J.; Sorge, C.; Ugus, O. A privacy model for smart metering. In Proceedings of the 2010 IEEE International Conference on Communications Workshops, Capetown, South Africa, 23–27 May 2010; pp. 1–5.
38. Kim, Y.; Ngai, E.C.H.; Srivastava, M.B. Cooperative state estimation for preserving privacy of user behaviors in smart grid. In Proceedings of the IEEE Second International Conference on Smart Grid Communications, (SmartGridComm 2011), Brussels, Belgium, 17–20 October 2011; pp. 178–183.
39. Erkin, Z.; Tsudik, G. Private computation of spatial and temporal power consumption with smart meters. In Proceedings of the 10th International Conference on Applied Cryptography and Network Security (ACNS 2012), Singapore, 26–29 June 2012; pp. 561–577.
40. McDaniel, P.D.; McLaughlin, S.E. Security and Privacy Challenges in the Smart Grid. *IEEE Secur Priv.* **2009**, *7*, 75–77. [CrossRef]
41. Munsing, E.; Mather, J.; Moura, S. Blockchains for decentralized optimization of energy resources in microgrid networks. In Proceedings of the IEEE Conferences on Control Technology and Applications (CCTA 2017), Mauna Lani Resort, HI, USA, 27–30 August 2017; pp. 2164–2171.
42. Luo, F.; Dong, Z.Y.; Liang, G.; Murata, J.; Xu, Z. A Distributed Electricity Trading System in Active Distribution Networks Based on Multi-Agent Coalition and Blockchain. *IEEE Trans. Power Syst.* **2018**, *34*, 4097–4108. [CrossRef]
43. Sahai, A.; Seyalioglu, H. Worry-free encryption: Functional encryption with public keys. In Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS 2017), Chicago, IL, USA, 4–8 October 2010; pp. 463–472.
44. Boneh, D.; Sahai, A.; Waters, B. Functional encryption: Definitions and challenges. In Proceedings of the Theory of Cryptography Conference, Providence, RI, USA, 28–30 March 2011; pp. 253–273.
45. Garg, S.; Gentry, C.; Halevi, S.; Raykova, M.; Sahai, A.; Waters, B. Candidate indistinguishability obfuscation and functional encryption for all circuits. In Proceedings of the 2013 IEEE 54th Annual Symposium on Foundations of Computer Science, Berkeley, CA, USA, 26–29 October 2013; pp. 40–49.
46. Bishop, A.; Jain, A.; Kowalczyk, L. Function-hiding inner product encryption. In Proceedings of the 21st International Conference on Advances in Cryptology (ASIACRYPT 2015), Auckland, New Zealand, 29 November–3 December 2015; pp. 470–491.
47. Abdalla, M.; Bourse, F.; De Caro, A.; Pointcheval, D. Simple functional encryption schemes for inner products. In Proceedings of the 18th IACR International Workshop on Public Key Cryptography (PKC 2015), Gaithersburg, MD, USA, 30 March–1 April 2015; pp. 733–751.
48. Datta, P.; Dutta, R.; Mukhopadhyay, S. Functional encryption for inner product with full function privacy. In Proceedings of the 19th IACR International Workshop on Public Key Cryptography (PKC 2016), Taipei, Taiwan, 6–9 March 2016; Springer: Berlin/Heidelberg, Germany, 2016; pp. 164–195.
49. Kim, S.; Kim, J.; Seo, J.H. A New Approach for Practical Function-Private Inner Product Encryption. *Theor. Comput. Sci.* **2019**, *783*, 22–40. [CrossRef]
50. Kim, S.; Lewi, K.; Mandal, A.; Montgomery, H.; Roy, A.; Wu, D.J. Function-hiding inner product encryption is practical. In Proceedings of the International Conference on Security and Cryptography for Networks (SCN 2018), Amalfi, Italy, 5–7 September 2018; pp. 544–562.
51. Diffie, W.; Hellman, M. New directions in cryptography. *IEEE Trans. Inf. Theory* **1976**, *22*, 644–654. [CrossRef]

52. Han, W.; Xiao, Y. Privacy preservation for V2G networks in smart grid: A survey. *Comput. Commun.* **2016**, *91*, 17–28. [CrossRef]
53. Vercauteren, F. Optimal pairings. *IEEE Trans. Inf. Theory* **2009**, *56*, 455–461. [CrossRef]
54. Barreto, P.S.; Naehrig, M. Pairing-friendly elliptic curves of prime order. In Proceedings of the International Workshop on Selected Areas in Cryptography, Kingston, ON, Canada, 11–12 August 2005; pp. 319–331.
55. A Portable and Fast Pairing-Based Cryptography Library (MCL). Available online: https://github.com/herumi/mcl/ (accessed on 29 January 2020).
56. GNU Multiple Precision Arithmetic Library (GMP). Available online: https://gmplib.org/ (accessed on 17 January 2019).
57. A Library for Doing Number Theory (NTL). Available online: https://www.shoup.net/ntl/ (accessed on 17 January 2019).
58. Aranha, D.F.; Karabina, K.; Longa, P.; Gebotys, C.H.; Hernandez, J.L. Faster explicit formulas for computing pairings over ordinary curves. In Proceedings of the 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques Advances in Cryptology (EUROCRYPT 2011), Tallinn, Estonia, 15–19 May 2011; pp. 48–68.
59. Sakemi, Y.; Nogami, Y.; Okeya, K.; Katou, H.; Morikawa, Y. Skew frobenius map and efficient scalar multiplication for pairing-based cryptography. In Proceedings of the 7th International Conference on Cryptology and Network Security, (CANS 2008), Hong Kong, China, 2–4 December 2008; pp. 226–239.
60. Jeon, S.Y.; Im, J.H.; Lee, M.K. Performance improvement of inner product encryption using parallel processing. In Proceedings of the 5th International Conference on Next Generation Computing 2019 (ICNGC 2019), Chiang Mai, Thailand, 19–21 December 2019; pp. 1–4.
61. Go-ethereum. Available online: https://github.com/ethereum/go-ethereum/ (accessed on 22 January 2020).
62. Buterin, V.; Reitwiessner, C. EIP197: Precompiled Contracts for Optimal Ate Pairing Check on the Elliptic Curve alt_bn128. Available online: https://eips.ethereum.org/EIPS/eip-197 (accessed on 30 January 2020).
63. Roiné, L.; Therani, K.; Manjili, Y.S.; Jamshidi, M. Microgrid energy management system using fuzzy logic control. In Proceedings of the 2014 World Automation Congress (WAC), Waikoloa, HI, USA, 3–7 August 2014; IEEE: Hoboken, NJ, USA, 2014; pp. 462–467.
64. Han, L.; Morstyn, T.; McCulloch, M. Incentivizing prosumer coalitions with energy management using cooperative game theory. *IEEE Trans. Power Syst.* **2018**, *34*, 303–313. [CrossRef]
65. Jiang, X.; Xiao, C. Household Energy Demand Management Strategy Based on Operating Power by Genetic Algorithm. *IEEE Access* **2019**, *7*, 96414–96423. [CrossRef]
66. Zeng, P.; Li, H.; He, H.; Li, S. Dynamic Energy Management of a Microgrid Using Approximate Dynamic Programming and Deep Recurrent Neural Network Learning. *IEEE Trans. Smart Grid* **2019**, *10*, 4435–4445. [CrossRef]
67. Zhou, S.; Hu, Z.; Gu, W.; Jiang, M.; Zhang, X.P. Artificial intelligence based smart energy community management: A reinforcement learning approach. *CSEE J. Power Energy Syst.* **2019**, *5*, 1–10. [CrossRef]
68. Farokhi, F.; Sandberg, H. Fisher Information as a Measure of Privacy: Preserving Privacy of Households with Smart Meters Using Batteries. *IEEE Trans. Smart Grid* **2017**, *9*, 4726–4734. [CrossRef]
69. Zhang, Z.; Qin, Z.; Zhu, L.; Weng, J.; Ren, K. Cost-Friendly Differential Privacy for Smart Meters: Exploiting the Dual Roles of the Noise. *IEEE Trans. Smart Grid* **2016**, *8*, 619–626. [CrossRef]
70. Chang, H.H.; Chiu, W.Y.; Sun, H.; Chen, C.M. User-Centric Multiobjective Approach to Privacy Preservation and Energy Cost Minimization in Smart Home. *IEEE Syst. J.* **2018**, *13*, 1030–1041. [CrossRef]