

Research article

Zero-knowledge machine learning models for blockchain peer-to-peer energy trading[☆]

Caixiang Fan^a, Amirhossein Sohrabbeig^b, Petr Musilek^{b,c}

^a Department of Computing Science, The King's University, 9125 50 St NW, Edmonton, T6B 2H3, Alberta, Canada

^b Department of Electrical and Computer Engineering, University of Alberta, Edmonton, T6G 2W3, Alberta, Canada

^c Department of Applied Cybernetics, University of Hradec Králové, 500 03 Hradec Králové, Czech Republic



ARTICLE INFO

Keywords:

Blockchain
Energy trading
Zero-knowledge machine learning
Privacy preservation
Smart grid
Internet of Things

ABSTRACT

Blockchain-based peer-to-peer energy trading enables individuals to directly share renewable energy using Internet of Things technologies. However, it faces significant challenges related to privacy, scalability, and the integration of advanced artificial intelligence. To address these issues, this article proposes zkPET, a secure and intelligent peer-to-peer energy trading framework. zkPET integrates machine learning and blockchain with advanced cryptographic techniques of zero-knowledge machine learning to protect user data while enabling intelligent decision making. In the zkPET framework, the computationally intensive operations of various machine learning models are executed off-chain, and only succinct cryptographic proofs of these computations are uploaded to the blockchain for verification and recording. In addition, a time-series clustering approach is incorporated into federated learning to enhance both inference accuracy and the efficiency of proof generation. Experimental validation using the zero-knowledge proof tool EZKL and a real-world electricity dataset demonstrates the feasibility and effectiveness of zkPET. The results underscore its potential to significantly improve privacy, scalability, and computational efficiency in decentralized energy trading, contributing to the advancement of secure and intelligent energy markets.

1. Introduction

The emergence and development of renewable energy resources, such as solar, wind, and hydrogen, and the Internet of Things have opened the door to peer-to-peer energy trading (P2P-ET). This allows individual peers to freely trade their excess energy with each other within their neighborhoods. Blockchain technology, due to its decentralization, immutability, and transparency, has been widely studied to build reliable and trustworthy P2P-ET platforms in smart grids [1–3]. Blockchain stores data in a distributed ledger, avoiding single points of failure and data tampering threats inherent in centralized databases. In addition, by eliminating intermediaries, it enables faster payment settlements and more cost-effective [4] transactions. Transparent trading policies are embedded in smart contracts that are executed on all nodes in the network and are publicly accessible.

However, blockchain-based P2P-ET faces significant challenges related to privacy, scalability, and integration of artificial intelligence (AI). First, blockchain's transparency raises privacy concerns for the energy trading participants. For example, if a user uploads original energy demand to the blockchain, it becomes easy for an adversary to launch linking attacks [5] through

[☆] This work has been partially supported by Sui Foundation through their Sui Academic Research Awards. Digital Research Alliance of Canada (alliancecan.ca) and Cybera (cybera.ca) partially supported this research work through their cloud computing resources.

* Correspondence to: 9125 50 St NW, Edmonton, AB, Canada, T6B 2H3 .

E-mail address: Stephen.Fan@kingsu.ca (C. Fan).

<https://doi.org/10.1016/j.iot.2025.101638>

Received 11 September 2024; Received in revised form 30 April 2025; Accepted 1 May 2025

Available online 28 May 2025

2542-6605/© 2025 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY-NC license (<http://creativecommons.org/licenses/by-nc/4.0/>).

data mining, especially in a community-based local energy market. This can result in user privacy breaches and even safety issues during zero energy demand. Second, blockchain's limited throughput poses significant scalability challenges for on-chain P2P-ET transactions. Third, blockchain's limited computational power and block size hinder the integration of advanced AI technologies. Smart contracts, with their high execution cost and restricted data types, are unsuitable for most AI algorithms. Therefore, it is difficult to achieve intelligent on-chain energy trading [6], such as residential load demand prediction [7,8], solar production forecasting [9–11], and optimization of the P2P-ET strategy [12].

Although contemporary technologies, such as encryption, differential privacy, and zero-knowledge proofs (ZKPs) [13–15], have been explored for privacy-preserving P2P-ET, existing solutions fail to fully integrate AI with privacy protection for blockchain-based energy applications. This gap motivates development of a scalable, privacy-preserving, and intelligent P2P-ET system.

In this paper, we explore blockchain-based peer-to-peer energy trading, focusing on both privacy protection and intelligence. Specifically, we aim to achieve intelligent trading through electricity demand time-series forecasting while preserving user's private energy data. We leverage zero-knowledge machine learning (ZKML) to build our solution. To the best of our knowledge, this is the first study to combine cutting-edge ZKML technology with blockchain in the energy sector. The main contributions of this paper can be summarized as follows:

- Proposal of a novel P2P-ET solution, called zkPET, which combines blockchain, federated learning (FL), and ZKML technologies to protect user privacy and enable on-chain intelligence for energy trading;
- Introduction of three potential models (or structures), namely centralized, decentralized, and federated ZKML, from the architectural perspective, and comparative analysis of their advantages and disadvantages;
- Development of dynamic time-series clustering algorithms by decomposing the data trend and seasonality to construct the federated ZKML model.
- Comprehensive evaluation of the proposed ZKML models on a real-world electricity consumption dataset to show insightful results.

The rest of this paper is organized as follows. Section 2 introduces preliminary cryptographic concepts of ZKP, polynomial commitment scheme (PCS), and ZKML. Section 3 briefly overviews related work on privacy-preserving energy trading and ZKML technologies. Section 4 formulates the problem and defines the model visibility. Section 5 details the design of centralized, decentralized, and federated ZKML models. Section 6 presents experimental performance evaluation results of the proposed solutions. In Section 7, we discuss some challenges, limitations, and potential solutions to improve the current zkPET system. Finally, Section 8 concludes the paper.

2. Background

This section introduces the background and preliminary concepts used in this paper. These include ZKP [16], KZG PCS [17], and ZKML.

2.1. Zero-knowledge proof

A mathematical proof is a compelling demonstration that convinces someone that a statement is true. A proof system refers to any procedure that determines what constitutes a convincing proof. Essentially, a proof system is specified by a verification procedure that evaluates any given statement alongside a claimed proof of its truth, ultimately determining the proof's validity [18]. In a proof system, there are usually two participants: a prover and a verifier. The prover generates the proof to convince the verifier that a statement is true. The verifier then validates the proof to determine whether to accept or reject the statement. A proof system typically has three basic properties: *completeness*, *soundness*, and *efficiency* [18]. *Completeness* means that any true statement should have a convincing proof of its validity. *Soundness* means that no false statement should have a convincing proof. *Efficiency* requires both the verification and proving procedures to be efficient.

A zero-knowledge proof [16] is a cryptographic proof in which the verifier learns nothing from the prover other than the validity of the statement being proven. Intuitively, a proof system is zero-knowledge if the verifier can compute the same outcome independently, without interacting with the prover [19]. Notably, generating zero-knowledge proofs is computationally intensive in reality, often much more costly than the original computation. However, proof verification is very cheap, which allows ZKP to be widely used in blockchain.

2.2. KZG polynomial commitment scheme

A commitment scheme is a cryptographic primitive that allows a prover to commit to a chosen statement m by producing a commitment c while keeping m hidden from others. The prover, or committer, can later reveal m , which the verifier can validate as truly corresponding to c . A commitment scheme requires two essential properties, *binding* and *hiding*. Binding ensures that once the commitment c is published, the committer cannot change m . Hiding ensures that c conceals the actual message m . A PCS leverages the powerful properties of polynomials to attain these features and offers additional advantages. First, it allows a committer to evaluate the polynomial $\phi(a) = b$ at a particular point a while keeping the polynomial $\phi(x)$ secret. Second, the commitment c typically has a smaller size than the polynomial itself, which is significant for on-chain verification.

The most famous and widely used PCS is the Kate–Zaverucha–Goldberg (KZG) scheme [17], named after its three inventors. KZG consists of four phases: setup, committing to the polynomial, proving an evaluation, and verification. During the setup phase, participants generate random values and perform a multi-party computation protocol to construct a one-time universal trusted setup (also known as a powers-of-tau ceremony) by computing $(g, g^\tau, g^{\tau^2}, \dots, g^{\tau^l})$ and releasing it publicly. Here, g is a generator of an elliptic curve group \mathbb{G} , l is the highest degree of the polynomials to be committed to, and τ is a secret parameter randomly chosen from the field element set \mathbb{F}_p . Then, in the second phase, a committer cannot directly compute a commitment $c = g^{\phi(\tau)}$ from a given polynomial $\phi(x) = \sum_{i=0}^l \phi_i x^i$ because τ is a secret value. Therefore, the committer borrows the setup results to compute the commitment c as

$$c = g^{\phi(\tau)} = g^{\sum_{i=0}^l \phi_i \tau^i} = \prod_{i=0}^l (g^{\tau^i})^{\phi_i}, \quad (1)$$

where ϕ_i is the coefficient of the given polynomial. In the third phase, the prover computes a proof $\pi = g^{q(\tau)}$ based on a given polynomial evaluation $\phi(a) = b$ and the quotient polynomial $q(x) = \frac{\phi(x)-b}{x-a}$, whose existence implicitly proves the former evaluation. The final phase is to verify the evaluation proof. This requires the verifier to check the validity of the following equation:

$$e\left(\frac{c}{g^b}, g\right) = e\left(\pi, \frac{g^\tau}{g^a}\right), \quad (2)$$

where e is a non-trivial bilinear mapping. Eq. (2) enables the verifier to validate that the quotient polynomial holds at τ without knowing the exact value of τ . Consequently, with high confidence, the verifier can conclude that the polynomial evaluation $\phi(a) = b$ is correct.

2.3. Zero-knowledge machine learning

ZKML combines the strengths of machine learning (ML), cryptography, and ZKPs to achieve intelligence and privacy protection within blockchain contexts. While it poses considerable challenges, its potential to transform data privacy and security is immense and warrants further exploration. Here, a ZKML proof refers to the inference step of an ML model rather than the model training.

EZKL [20] is a popular ZKML tool based on the KZG PCS and the well-known Halo2 ZKP system. Poseidon [21] cryptographic hash function, a ZK-friendly hash algorithm, is used to improve the hashing efficiency in EZKL. Written in Rust and wrapped with Python and JavaScript, EZKL offers user-friendly APIs. After proper setup, a prover can use EZKL to easily prove the correct inference of a public/private neural network on some private/public data, producing specific outputs. The correctness of this inference can be efficiently verified on-chain. A recent benchmarking report [22] shows that EZKL outperforms other two ZKML frameworks, RISC Zero [23] and Giza's Orion [24], in terms of proving time and memory usage. This demonstrates the effectiveness and efficiency of adopting EZKL to build our solution in this paper.

3. Related work

There have been numerous studies on privacy-preserving energy trading using various technologies. For example, some researchers combine differential privacy (DP) [25] with blockchain to hide the energy trading information [26–28]. Others leverage secure multi-party computation (SMC) [29] with blockchain to enable decentralized energy trading, allowing power suppliers to calculate consumer bills without accessing their private power consumption data [30,31]. Homomorphic encryption (HE) [32] is also employed with blockchain to protect privacy in energy trading and smart grid data management [33–36]. Recently, Lu et al. [37] integrated a trusted execution environment (TEE) [38] with a blockchain-based system to achieve privacy preservation in energy trading.

However, these techniques have limitations when applied to efficient, trustless, and intelligent peer-to-peer energy trading. SMC requires synchronized communication, which may not accommodate dynamic user joining or exiting. HE is extremely expensive, especially for computationally intensive applications such as machine learning. DP only provides provable privacy without verifiability. TEE relies on isolated and trusted hardware, introducing additional costs [39].

Compared to these techniques, ZKPs offer unique advantages in privacy preservation and computation verification [39]. For instance, Pop et al. [13] used ZKPs to develop a privacy-preserving energy demand response (DR) program on a public blockchain. The proposed solutions integrate ZKPs with the DR smart contracts to hide the aggregated energy data of prosumers. ZGridBC [14] aggregates smart meter records with evidentiality using ZKP in smart grids. Another work [15] shares a similar idea by proposing a privacy-preserving energy trading mechanism using public blockchain and ZKPs. It employs the Pedersen Commitment [40] protocol to hide the original bid amount in energy trading.

ZKP technology not only protects privacy but also makes the computation verifiable. However, previous solutions did not consider intelligence in energy trading. In this paper, we fill this gap by leveraging ZKML to prove the computation graph of a simple neural network for energy demand forecasting, which achieves verifiable and intelligent trading while preserving user privacy, as shown in Table 1.

Table 1
A detailed comparison between this study and related works

Work	Technique	Use case	Privacy	Verifiable	Intelligent
[26–28]	DP	ET	✓	✗	✗
[30,31]	SMC	ET	✓	✗	✗
[34–36]	HE	Smart Grids	✓	✗	✗
[37]	TEE	ET	✓	✗	✗
[13]	ZKPs	DR	✓	✓	✗
[14]	ZKPs	Smart Grids	✓	✓	✗
[15]	ZKPs	P2P-ET	✓	✓	✗
This study	ZKML	P2P-ET	✓	✓	✓

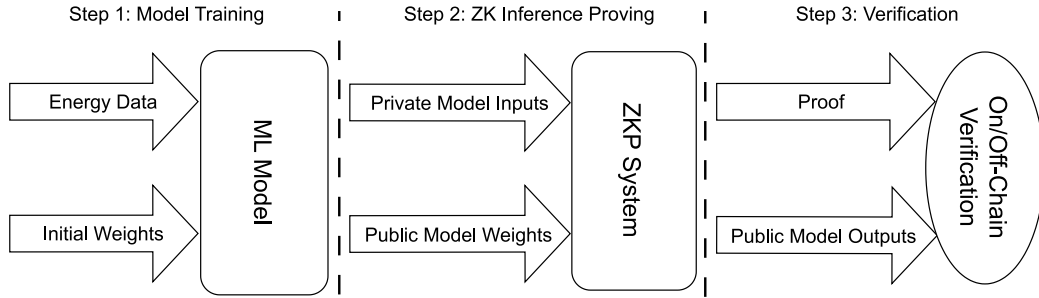


Fig. 1. Three typical steps of ZKML.

4. Problem definition

In P2P-ET, a consumer is an individual residence, called a site, that only consumes electricity without any generation capacity. In contrast, a prosumer is equipped with renewable energy generators, such as solar photovoltaic roofs and wind turbines. An energy aggregator is an entity, whether a person, business, or technology, that bundles DERs from sites to participate in larger electricity markets as a single entity, known as a virtual power plant (VPP). Aggregation offers lower prices and transaction costs for consumers and greater energy DR flexibility compared to individual households.

It typically involves three steps in applying ZKML to on-chain trading, including model training, ZK inference proving, and verification, as shown in Fig. 1. In this design, a prover can be a consumer/prosumer or an energy aggregator. They use their computational resources and a specific proof system to prove a statement, such as claiming that they ran a publicly available neural network on some private data and it produced a particular output. All the statement elements, including the instance data, e.g., a hash of the model input and actual inference output, and the bytes of the cryptographic proof, are contained in a proof file. A verifier could be a program or a smart contract containing a verify function, which takes the proof bytes and instance data as inputs and efficiently validates the correctness of the statement without actually executing the intensive computation.

In a ZKML system, a preliminary task is to set the visibility of each model element, which determines the statement we are proving in the arithmetic circuit settings. Similar to the boolean circuits with logical operations such as AND and OR, arithmetic circuits in ZKP are a model for computing polynomials, with arithmetic operations, such as addition and multiplication. Here, we design zkPET’s visibility as

- Private model input, which means that the historical time-series energy data used for model inference is only visible to the prover, defined by the *input_visibility* configuration;
- Fixed model parameters, which indicate that the model weights are fixed and public at setup, visible to the prover, and only committed to the verifier in the verifying key, defined by the *param_visibility* configuration;
- Public model output, which means that the inferred future time-series energy data is part of the proof file and shared with the verifier, defined by the *output_visibility* configuration.

Given the visibility design, i.e., private input, fixed (or public) weights, and public output, and the designated proof system Halo2, the prover needs to prove that it knows some secret X such that

$$f(X; \omega, b) = Y, \tag{3}$$

where f is the publicly visible relation in the form of a trained neural network model, X is the input, Y is the output, ω is the weight vector, and b is the bias vector. The KZG commitment is used by default to ensure that the prover does not change the input data in a particular ZKML process.

For a time-series forecasting task, the prover has a private input vector $x \in \mathbb{R}^L$, representing a historical time series with length L , and wants to predict T future time steps $y \in \mathbb{R}^T$. The prover first generates a KZG commitment c_x . Then, given a public ML

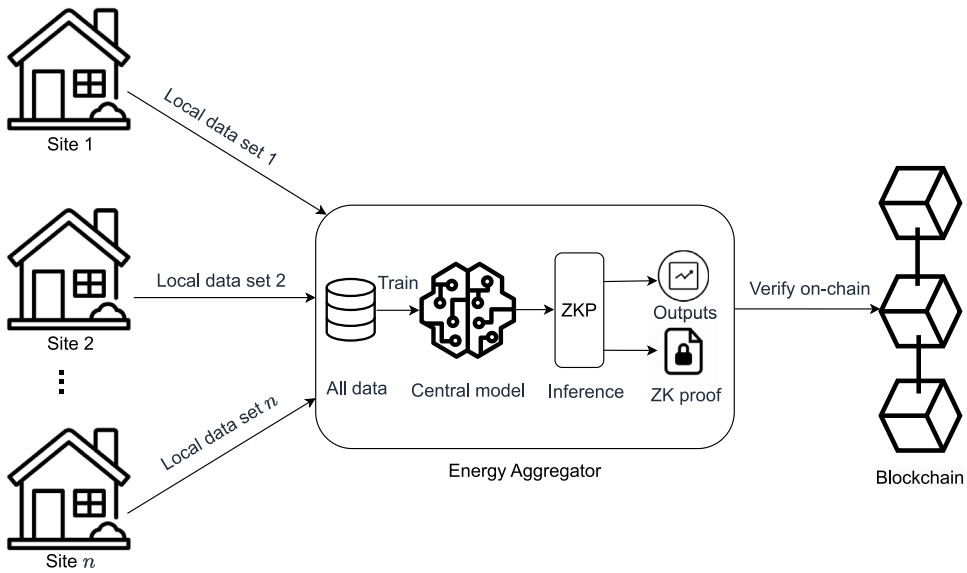


Fig. 2. A centralized ZKML model for P2P-ET.

model with weights ω represented as $f_\omega : \mathbb{R}^L \rightarrow \mathbb{R}^T$, the prover needs to create a proof π that states the prover faithfully executes the model with private input x and obtains a forecasting output $y = f_\omega(x)$.

5. ZKML solutions for P2P-ET

Applying ZKML in energy trading requires a systematic approach to resource allocation for each step in the process. This involves determining which components are responsible for training the model and/or proving the inference, and how many proofs need verification. As a result, three distinct ZKML models have been developed based on their system structures: centralized, decentralized, and federated ZKML.

5.1. Centralized ZKML

The centralized ZKML model requires the energy aggregator to collect energy data from all individual sites. Then, a central model is trained on this data collection, and the inference is proved through a ZKP system on the aggregator server, as shown in Fig. 2. This solution offers a straightforward strategy, resulting in a single cryptographic proof, which makes on-chain verification inexpensive. In addition, it moves all computation loads, such as model training and ZKML inference proving, to the server side, thus eliminating the need for local devices to have high computational power. This makes the solution more IoT-friendly.

However, despite simplifying the process, the centralized ZKML approach has three notable disadvantages. First, the transmission of energy data between local sites and the aggregator raises privacy and security concerns. While encryption techniques can address this issue, there remains a small time window during which adversaries can attack when decrypting the data on the server side. Second, the data transmission requires significant bandwidth and stable connections to prevent network congestion and data losses. Finally, as the number of individual sites increases, the number of parameters in the central model grows linearly. This linear growth in the constraints of the circuits for ZKML renders the model remarkably unscalable.

5.2. Decentralized ZKML

The decentralized ZKML model requires local sites to train their own ML models and then prove the inference using ZKP systems. All proofs can be sent to the aggregator and aggregated through recursive proving before on-chain verification, as shown in Fig. 3. Local sites also have the option to directly create a verifier from the local proof and deploy it to the blockchain. This approach significantly enhances data privacy and reduces the communication bandwidth requirements because a zkSNARK [41] proof is succinct and much smaller than the raw or encrypted data.

However, this model presents other concerns. First, local sites must have significant computation resources to train a model and prove the inference process. Second, if a large number of local sites choose not to aggregate their proofs and instead deploy their verifiers to the blockchain, the verification stage will be very costly in terms of gas fees, resulting in poor proof verification scalability.

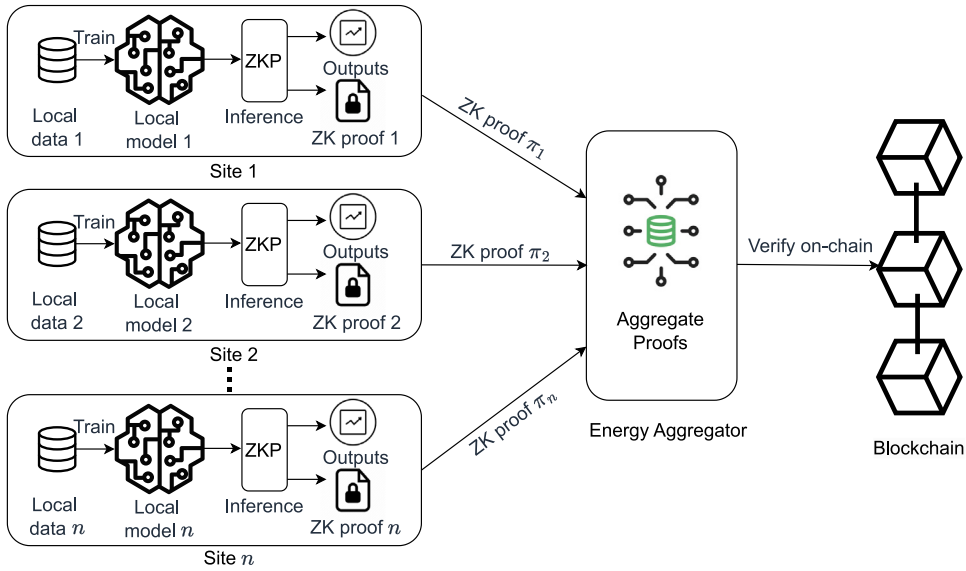


Fig. 3. A decentralized ZKML model for P2P-ET.

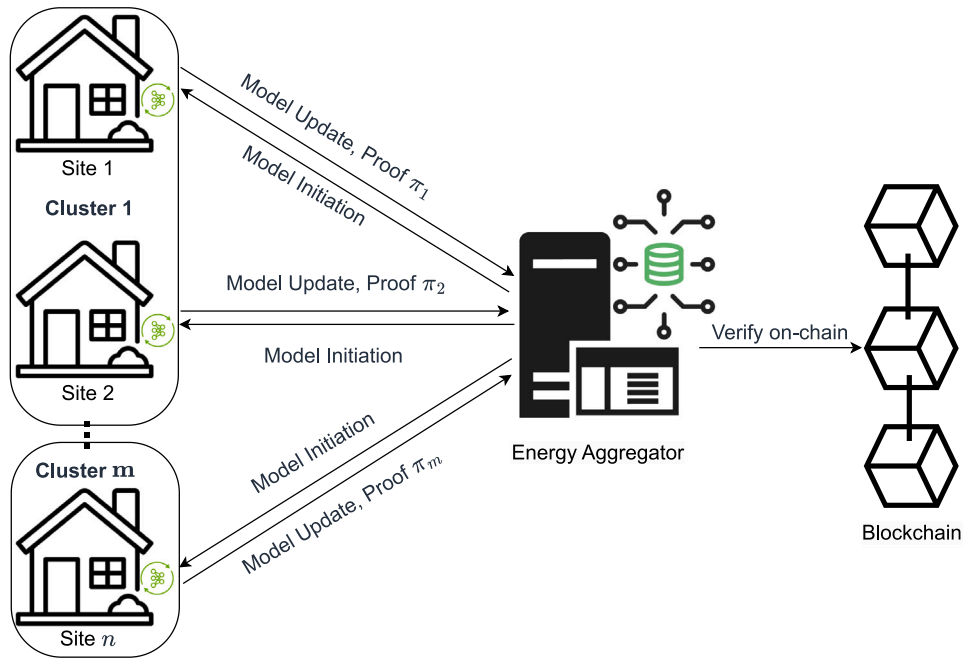


Fig. 4. A federated ZKML model for P2P-ET.

5.3. Federated ZKML

To address the issues of privacy, proving, and verification scalability in the aforementioned approaches, we introduce a clustered federated ZKML model, as shown in Fig. 4. In this method, we use FL techniques, especially the Flower [42] framework, to protect data privacy by collaboratively training ML models on local sites without transferring their sensitive energy data. Compared to traditional energy markets, peer-to-peer energy trading enables a more dynamic and flexible pricing mechanism, allowing buyers and sellers to negotiate mutually beneficial rates. This decentralized market structure incentivizes self-interested participants to collaborate on training FL models, thereby contributing to overall system intelligence. Moreover, the proposed zkPET framework has a strong potential to further enhance participant engagement through token-based incentives on the blockchain, effectively aligning individual objectives with collective system performance.

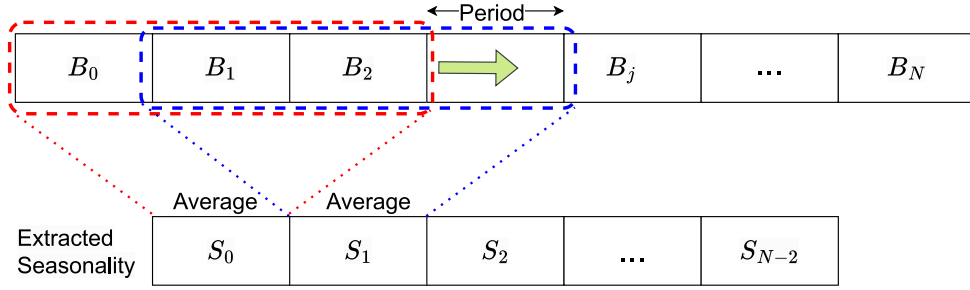


Fig. 5. Seasonality extraction using a sliding window. In this example, the sliding window spans three seasonal periods, with a stride of one.

According to the visibility definition in Section 4, these models will be publicized after training. In the centralized and decentralized ZKML models, this public access allows adversaries to carry out reverse engineering attacks because their model weights are explicitly linked to the energy data of specific sites. In federated ZKML, the FL aggregation mechanism, i.e., weighted average, effectively protects the model from this type of attack. In addition, federated ZKML can significantly reduce the number of models that need to be publicized compared to the decentralized ZKML.

However, due to the heterogeneity of local datasets across different individual sites, it is challenging for a collaborative FL model to capture accurate patterns. For example, in a time-series load forecasting task, an FL model may struggle to effectively learn the demand patterns of all users, as their electricity usage patterns can vary significantly from one another [43]. To address this issue, we propose a time-series clustering algorithm that decomposes the trend and seasonality features of local datasets and combines these components to calculate the distances between two time-series datasets.

5.3.1. Time-series data clustering

The initial step in the time-series clustering process involves extracting trend and seasonality. This step employs a decomposition approach similar to the seasonal-trend decomposition using the Loess (STL) [44] method. In the proposed method, the trend component is first extracted using a moving weighted average. This function assigns more weight to closer neighboring data points and less to those farther away. The extracted trend pattern is then subtracted from the original input to yield the detrended time series, which is subsequently used to extract seasonality.

To draw out seasonality from the detrended time series, we first determine the period of seasonality. Expert knowledge indicates that the hourly energy consumption of households exhibits daily and weekly seasonal patterns, corresponding to 24 and 168 h, respectively. These periodicities are well-established in the literature and reflect typical behavioral cycles in residential settings, such as daily routines and weekend effects [45]. Then, a convolution-like approach is applied, where a window with a length of n seasonal periods and a stride of m periods is passed over the data. These parameters were selected based on the trade-off between temporal resolution and noise reduction. At each step, the average of the n periods that the window encompasses is considered to be the seasonality component of the corresponding segment, as illustrated in Fig. 5.

The following equation defines the function applied at each step

$$S_i = \frac{1}{n} \sum_{j=i}^{i+n-1} B_j, \quad (4)$$

where B_j represents the j th block of the detrended data. Each block, B_j , is a vector containing p data points. These data points span indices from $i \cdot p$ to $(i + 1) \cdot p - 1$. In this context, S_i represents the average of n consecutive blocks of data starting at block i . The sum includes all data points from blocks B_i to B_{i+n-1} .

It is crucial to ensure that the extracted trend and seasonality components are accurate representations of the time-series characteristics while being concise enough for efficient distance metric computation. To achieve this, the extracted trend pattern is downsampled, which also helps protect the original sensitive energy data from leakage. For the seasonality component, an appropriate window size and stride are selected to maintain a balance between representation accuracy and computational efficiency. This decomposition approach effectively isolates the inherent characteristics of the time series, enabling a more accurate measurement of similarity between different series.

The second step is to measure the distance between any two varieties based on the decomposition results. The dynamic time warping (DTW) [46] distance metric is used for this purpose due to its high robustness in handling time-series data with non-linear alignments. DTW allows flexible matching by aligning similar patterns in the time series, regardless of their specific timings. To compute the similarity between time series, we measure the DTW distances between both trend and seasonality components separately, and combine them using the Euclidean norm. This assumes equal importance for both components and captures their independent contributions to the overall similarity.

The third step is to group the time series with the obtained short representations and calculated distances. The k-Medoids [47] clustering algorithm is selected for this task. Compared to k-Means, the k-Medoids algorithm is less sensitive to noise and outliers, making it more suitable for time-series data, which often contain anomalies.

Algorithm 1 Time-series Clustering Algorithm**Input:** Time series data $\{X_i\}_{i=1}^N$, seasonality periods P_1, P_2 **Output:** Clusters $\{C_j\}_{j=1}^K$

```

1: Pattern Extraction:
2: for each time series  $X_i$  do
3:   Extract trend component  $T_i$  using a moving weighted average with tricube weights
4:   Detrend the series:  $\hat{X}_i = X_i - T_i$ 
5:   Extract seasonality component  $S_i$ 
6:   for each seasonality period  $P$  in  $\{P_1, P_2\}$  do
7:     Apply the convolution-like approach with window length  $n \times P$  and stride  $m \times P$ 
8:     Compute seasonality component  $S_i$  as the average of the  $n$  periods
9:   end for
10:  Downsample the trend pattern  $T_i$  for efficiency and privacy
11: end for
12: Distance Measurement:
13: Initiate an empty distance matrix  $D_{N \times N}$ 
14: for each time series pair (i,j) do
15:   /* Combine DTW distances of trend and seasonality */
16:   Calculate their distance  $d = \sqrt{DTW(T_i, T_j)^2 + DTW(S_i, S_j)^2}$ 
17:    $D_{i,j} = d$ 
18: end for
19: Clustering:
20: Apply the k-Medoids clustering algorithm to the distance matrix  $D$ 
21: Determine the optimal number of clusters  $K$  using the elbow method and the silhouette metric

```

The optimal number of clusters is determined using the elbow method and the silhouette metric [48]. The clustering process is summarized in Algorithm 1.

Furthermore, several considerations regarding the clustering algorithm must be taken into account. First, due to the non-stationarity of the underlying data distribution, the relationships between different channels (i.e., the distinct variables or features observed over time in a multivariate time series) may change over time. Therefore, the clustering algorithm should be run regularly at specific intervals to continually adapt to these changes. We must also consider the possibility of households joining or leaving the system at any time. If such events occur, an iteration of the clustering algorithm must be executed to accommodate the changes. This dynamic clustering procedure outlined in Algorithm 2, ensures the relevance and accuracy of the clustering results despite potential changes in the dataset composition.

Algorithm 2 Dynamic Clustering Algorithm**Input:** Regular interval T

```

1: Regular Adaptation:
2: while true do
3:   Wait for interval  $T$ 
4:   Re-run Algorithm 1 to adapt to any changes in data distribution
5:   Publish the clustering results  $\{C_j\}_{j=1}^K$  to the cloud
6:   Check for new or missing households
7:   if new or missing households are detected then
8:     Re-run Algorithm 1 with updated data
9:     Publish the updated clustering results  $\{C'_j\}_{j=1}^K$  to the cloud
10:  end if
11: end while

```

5.3.2. Federated model training

The clustered time series approach ensures that all sites within a cluster share similar energy data patterns. This strategy allows to aggregate the model weights using the basic FedAvg algorithm within a cluster without sacrificing accuracy and generalization. Given the clustering results with m disjoint clusters (C_1, C_2, \dots, C_m) , FL is leveraged to train a homogeneous ML model for each cluster $C_i, 1 \leq i \leq m$. This design enables multiple central model aggregation processes to run in parallel, improving the learning efficiency of the entire system, as illustrated in Fig. 4.

To start training the FL model, the aggregator server initiates model parameters by randomly assigning weights to the local model. Then, local machines train the model on the private local datasets and iteratively upload their model weights to the central

server until the algorithm converges. In each iteration, the central server aggregates the model weights using a common strategy, federated averaging (FedAvg), which calculates the weighted average of all received weights based on the number of training data instances at each site. An iteration ends with the central server updating the parameters to the local sites.

For simplicity, we employ the popular FL framework Flower with the widely-used FedAvg aggregation algorithm for the central model. For a particular cluster C_1 with S sites, the learning objective is represented as

$$\min_{\omega} F(\omega) = \sum_{j=1}^S \frac{n_j}{n} F_j(\omega), \quad (5)$$

where $F_j(\omega)$ is the local loss of site j , n_j is the number of training data instances at site j , and n is the total number of training instances across all clients in C_1 . Thus, $\frac{n_j}{n}$ represents the weight of site j in FedAvg. Once the training is complete, the model is published to the cloud, making it publicly visible.

5.3.3. Inference proving

Given regularly updated clustering results $\{C'_j\}_{j=1}^K$ and the associated trained models $\{M_j\}_{j=1}^K$, EZKL can be used to prove the model inference. EZKL utilizes a zkSNARK proof system to verify the correctness of computational steps. The proposed model depends on the robustness and integrity of this proof system, which is thoroughly explained by Vitalik Buterin, co-founder of the Ethereum blockchain, in his detailed article on the arithmetic mechanisms [49]. Since EZKL requires the ML model to be in the Open Neural Network Exchange (ONNX) format, the prover needs to create an ONNX file based on the obtained ".pth" checkpoint model and the model input data x , which contains L historical energy data samples for prediction. As x is sensitive, it should be serialized into a JSON file on a local site. Then, the prover generates and calibrates the circuit configurations in a JSON file. This settings file contains all necessary circuit parameters, such as the visibility, logarithmic rows of the lookup table, number of constraints, and model instance shapes. After retrieving a public Structured Reference String (SRS) file generated by the power-of-tau ceremony, the model is compiled into a format ready for proving.

Given the compiled model M_{compiled} , the downloaded KZG SRS file F_{srs} , and the calibrated settings $S_{\text{calibrated}}$, the prover can use EZKL's setup function G to generate the proving key P_{key} and the verifying key V_{key} as

$$G(M_{\text{compiled}}, F_{\text{srs}}, S_{\text{calibrated}}) \Rightarrow (P_{\text{key}}, V_{\text{key}}). \quad (6)$$

The next step is to generate the witness W , a collection of the computed intermediate values, called advice values, and the private inputs in the Halo2 proof system. Then, a prover creates a proof π containing the model output on a local site as

$$P(P_{\text{key}}, M_{\text{compiled}}, F_{\text{srs}}, W) \Rightarrow \pi. \quad (7)$$

5.3.4. Proof aggregation and verification

Proof aggregation is a technique for reducing the proof size or resolving the verification scalability issues. Two popular proof aggregation schemes are SnarkPack [50] of Groth16 proofs and aPlonK [51] for PlonK proofs. The basic idea of aggregation is to recursively compute a large proof or multiple proofs and generate a final smaller proof as

$$AGGR(P_{\text{key}}, F_{\text{srs}}, \pi_1, \pi_2, \dots, \pi_m) \Rightarrow \bar{\pi}, \quad (8)$$

where m individual proofs collectively play the role of witness in a prove function. This aggregation proves that the prover knows the m witness proofs such that the aggregation statement is valid. In zkPET, this technology is used to aggregate all local proofs into one proof on the energy aggregator. It should be noted that the aggregation process is computationally expensive and requires more powerful machines with graphics processing units (GPUs), especially when m is large. Then, this proof is sent to the verifier and gets verified as

$$V(V_{\text{key}}, S_{\text{calibrated}}, F_{\text{srs}}, \bar{\pi}) \Rightarrow \text{true/false}. \quad (9)$$

Since the proof is succinct, this step can be moved on-chain by deploying a verifier smart contract and providing the proof bytes as the inputs of its `verifyProof` function, as shown in Fig. 6. Once successfully verified, the final proof mathematically implies the correctness of all original statements.

5.4. System process flow

Fig. 7 illustrates the detailed system process flow of the proposed solution. When an energy trading transaction is initiated, the system checks whether both privacy and intelligence are required. If both are required, the process will proceed through the intelligent subsystem (on the left of Fig. 7) and then through the ZKP subsystem (in the center of the figure). In this contribution, we focus on the implementation of the intelligence and ZKP subsystems, concluding with the on-chain verification of model inference proofs. More details about the energy trading subsystem on the blockchain are discussed in our previous article [2]. It is important to note that ML model training is an asynchronous process in relation to energy trading. This means that training can run regularly in the background on different machines, following the three proposed model structures.

A brief comparison of the three ZKML models for P2P-ET applications is shown in Table 2. Even though the federated ZKML is more complicated in implementation, it theoretically provides better data privacy and higher system scalability.

```

contract Halo2Verifier {
    function verifyProof(
        bytes calldata proof,
        uint256[] calldata instances
    ) public view returns (bool) {
        assembly {
            // Read EC point (x, y) at (proof_cptr, proof_cptr + 0x20),
            // and check if the point is on affine plane,
            // and store them in (hash_mptr, hash_mptr + 0x20).
            // Return updated (success, proof_cptr, hash_mptr).
            function read_ec_point(success, proof_cptr, hash_mptr, q)
                -> ret0, ret1, ret2
            {
                let x := calldataload(proof_cptr)
                let y := calldataload(add(proof_cptr, 0x20))
            }
        }
    }
}

```

Fig. 6. A snapshot of the sample Halo2 verifier contract generated by the EZKL tool.

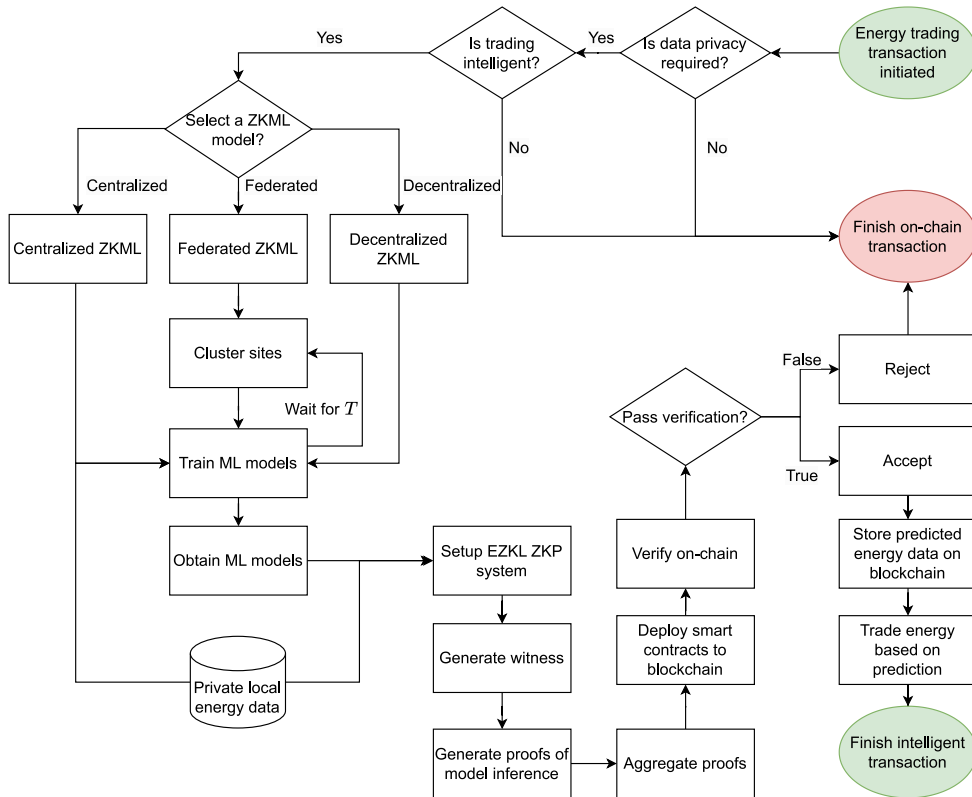


Fig. 7. A detailed system process flow diagram of the proposed methodology.

6. Performance evaluation

A very simple one-layer linear long-term time-series forecasting (LTSF) model, known as LTSF-Linear [52], is selected as the basic ML model to evaluate the proposed structures. This model is ideal for ZKML due to its simplicity and has been verified to outperform more complex Transformer-based solutions in energy consumption time-series forecasting tasks [52]. The dataset, which has been

Table 2
A brief comparison of the three ZKML models.

Models	Simplicity	Data privacy	Scalability
Centralized ZKML	High	Low	Low
Decentralized ZKML	High	High	Low
Federated ZKML	Medium	High	High

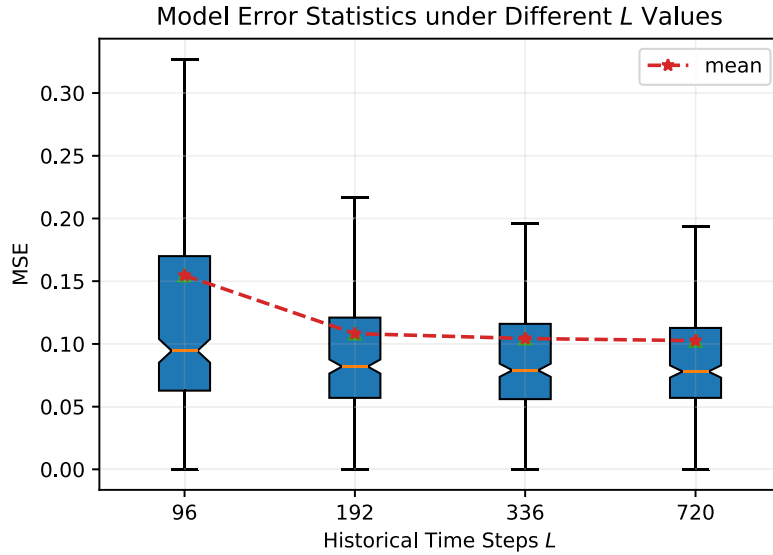


Fig. 8. LTSF-Linear MSE statistics of different historical L time steps.

widely used in time-series forecasting studies, contains 321 variables representing the real-world hourly electricity consumption of individual residents (sites) between 2012 and 2014. All research artifacts, including raw data, source code, software tools, and visualization plots, are publicly available on [GitHub](#). All experiments are conducted on a cloud server with 8 virtual CPUs (Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20 GHz) and 16 GB of RAM. The Mean Squared Error (MSE) is used as the performance evaluation metric in the model training stage. In the ZKML inference proving stage, the EZKL v10.0.2 tool is run with its optimized Halo2 proof system to evaluate the following performance metrics:

- Proving time: the time used to generate a proof for a given model;
- CPU usage: the number of CPU cores used during the proof generation process;
- Memory usage: the maximum amount of memory used during the proof generation process.

6.1. Model training

In P2P-ET, the day-ahead market is very popular, where participants trade energy and finalize the price 24 h before the actual energy delivery. Therefore, we select the forecasting future time-series sequence length $T = 24$ throughout our experiments. For a time-series model, a short historical sequence length L cannot provide enough information to predict the future, while too large L values can lead to overfitting. Thus, values $L = 96, 192, 336, 720$, representing 4, 8, 14, and 30 days, respectively, are selected to train the model and determine the best configuration.

The linear models for 321 individual sites are trained using their local data. From the MSE statistics in [Fig. 8](#), it is evident that $L = 192$ provides the best performance in terms of error reduction, as its further increases L does not improve performance. The other two types of models, DLinear and NLinear, exhibit similar error statistics in the experiments. Thereby, we select $L = 192$ as the default configuration.

To better understand how the model size influences the performance of each structure, the number of model parameters of different types of models with varying input sizes are listed in [Table 3](#). As shown, Linear and NLinear models have the same number of parameters, while DLinear models have twice as many parameters for all L configurations.

6.2. Time-series clustering results

Algorithm 1 was run for all 321 time series, testing the number of clusters k from 2 to 20. The range of k values was selected to balance computational efficiency and interpretability: while theoretically up to 321 clusters could be considered, clustering with

Table 3
Number of model parameters for a single site.

Models	$L = 96$	$L = 192$	$L = 336$	$L = 720$
Linear	2328	4632	8088	17 304
NLinear	2328	4632	8088	17 304
DLinear	4656	9264	16 176	34 608

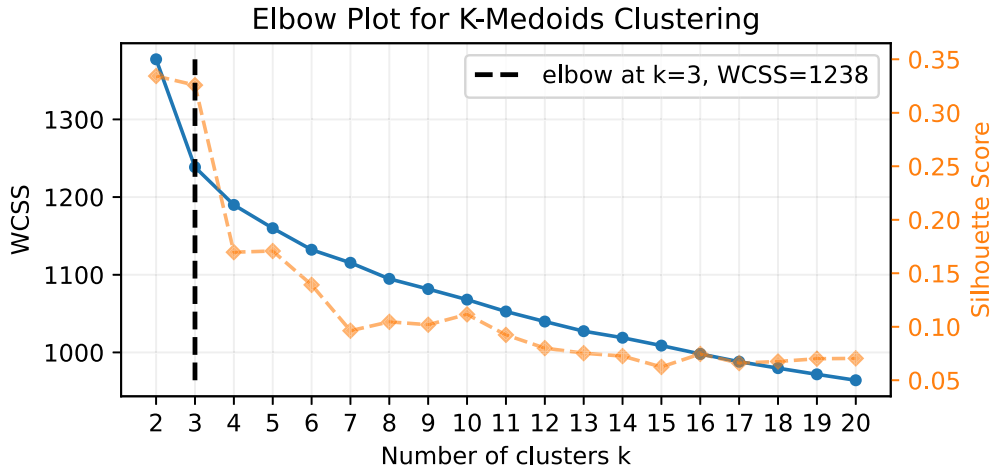


Fig. 9. Time-series clustering convergence against the cluster numbers from 2 to 20.

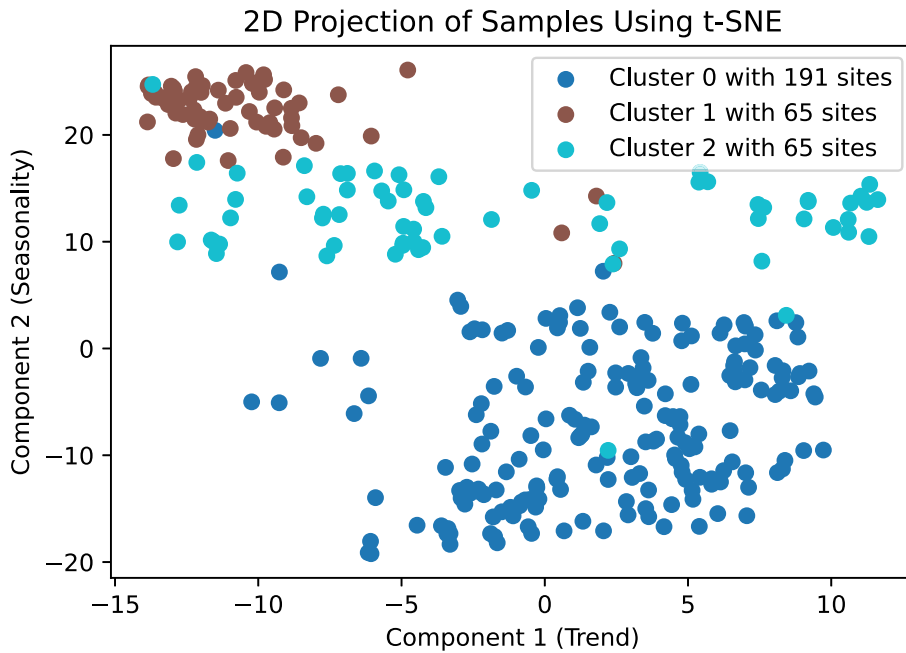


Fig. 10. Result of clustering time series in 2D space projected by t-SNE.

such large k would likely overfit noise rather than reveal meaningful patterns [53]. The results are shown in Fig. 9. Using the elbow method, the optimal number of clusters corresponds to the elbow point at $k = 3$. This value of k also yields relatively high silhouette scores. Therefore, the k-Medoids algorithm is used to classify all sites into three clusters. Fig. 10 depicts the clustering results of the time series projected onto 2D space using t-SNE [54].

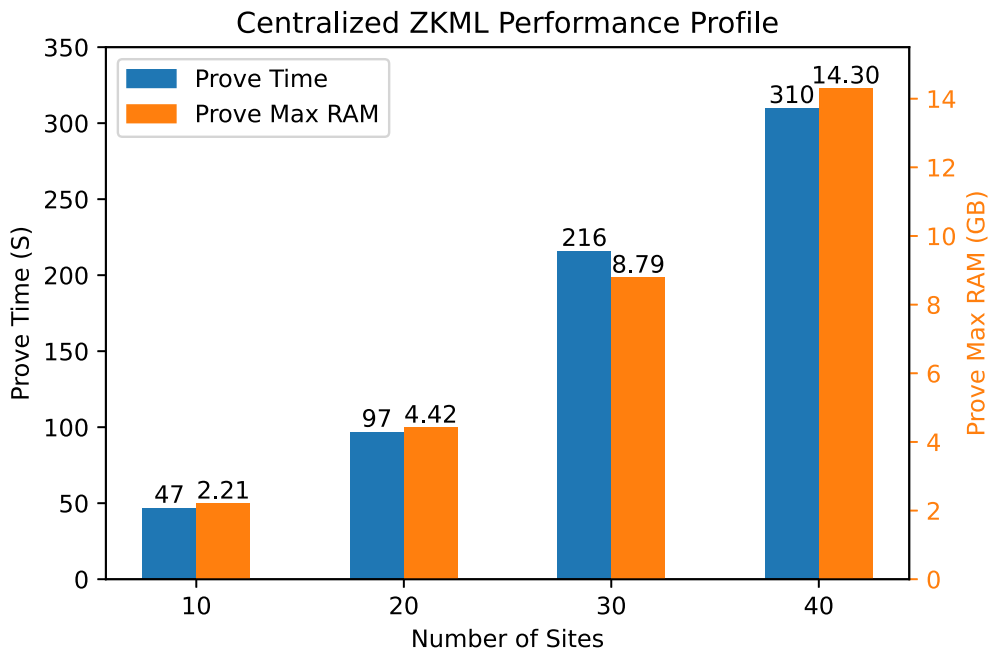


Fig. 11. Centralized ZKML proving performance profiling.

6.3. Proving performance profiles

Proving is the most computationally expensive step in the ZKML process. Therefore, the experimental evaluation focuses on the performance of generating proofs for the three proposed models.

6.3.1. Centralized ZKML proving performance

The scalability of centralized ZKML is assessed by progressively training centralized models using the first i sites, with i increasing in a step of 10. Each centralized model comprises a module list of i individual LTSF-Linear models with $L = 192$, trained on the central server with the associated central dataset. Next, the performance of inference proving is evaluated with results shown in Fig. 11.

The findings indicate that this approach scales up to 40 sites with the current setup with 16 GB RAM. However, both proving time and maximum memory usage increase linearly, from 47 s and 2.21 GB for 10 sites to 310 s and 14.30 GB for 40 sites, respectively. This upward trend results in the model-proving process for 50 sites exhausting memory and being terminated. These observations highlight that the centralized ZKML requires a more powerful computational setup and exhibits limited scalability under current conditions.

6.3.2. Decentralized ZKML proving performance

For decentralized ZKML, proofs are aggregated to improve the verification scalability and reduce on-chain verification costs. Therefore, its scalability is profiled by comparatively evaluating its proving performance with and without aggregation, with results shown in Fig. 12.

The model proving process was run for all 321 Linear models with $L = 192$ using the EZKL tool, collecting their performance of proving time, CPU usage, and maximum RAM for statistics. The box plot on the left side of each metric indicates the performance statistics for a single proof, while the right plot represents the statistics of proofs for aggregation, generated by the EZKL command `prove` with proof type `for-aggr`.

The percentage of CPU reports core utilization with respect to a single core. In Fig. 12, 450% means that the proving job used about 5 cores out of 8 cores in the server. This is beneficial from the Halo2 proof system's use of Rayon, a Rust data-parallelism library that enables the `multicore` feature through parallelism, dramatically upgrading circuit execution performance. However, this parallelism has not been fully incorporated into EZKL's implementation of proof aggregation. Consequently, single-proof generation without aggregation shows much higher CPU core usage than proof generation with aggregation.

Proving memory usage is closely related to the size of the downloaded SRS file, determined by the trusted setup parameter `logrows`. A higher `logrows` value results in a larger SRS file and increased memory consumption during proof generation. In the experiments, a single proof has a calibrated setting of `logrows = 12` and a 513 KB SRS file, while an aggregate proof uses 23 `logrows` and an SRS file of 1.1 GB. This explains why the maximum RAM usage of the aggregate proofs is much higher than that of the single proofs without aggregation, as shown in Fig. 12.

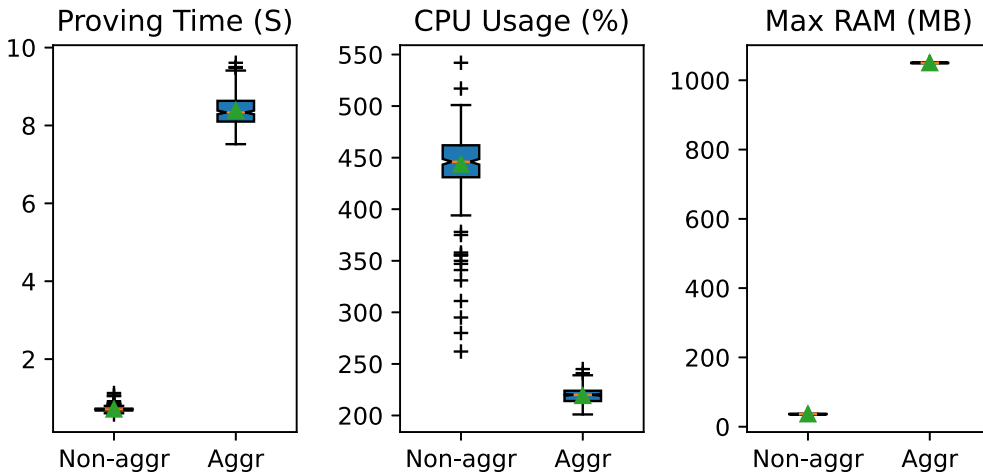


Fig. 12. Decentralized ZKML proving performance profiling.

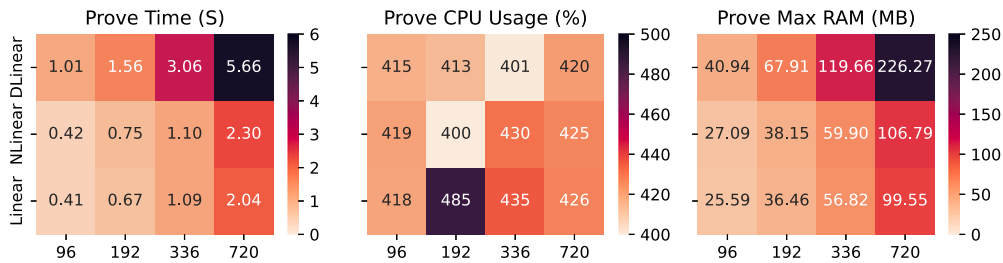


Fig. 13. Single site ZKML proving performance against different models with various historical sequence lengths.

To explore the influence of different linear model types (Linear, DLinear, and NLinear) and varying L values on proof generation performance, site 40 is selected for evaluation. Fig. 13 presents heat maps of model inference proving performance, including proving time, CPU usage, and maximum memory usage across different model types and historical sequence length configurations. Here, darker colors represent higher values.

The Linear models with $L = 96$ and $L = 192$ achieve sub-second proving times, while larger L values for NLinear or DLinear models require significantly more time to generate proofs. Interestingly, all configurations have a very similar proving CPU usage, averaging around 4.2 cores, due to the similarity of their model structures.

The optimal historical sequence choice, $L = 192$, for the Linear model, consumes only 36.46 MB of memory, approximately half of the DLinear model’s 67.91 MB memory usage under the same configuration. This aligns with the proportional relationship in the number of model parameters, as the DLinear model has twice the parameters of the Linear model.

6.3.3. Federated ZKML proving performance

Based on the clustering results in Section 6.2, the last cluster with 65 sites is selected to profile the ZKML model proving performance. Before proving the inference, Flower [42] FL framework is used to train a Linear model for the cluster. After training, this model is published to the cloud for public access. Then, each site downloads the model to prove that they execute the model’s computation based on their private energy data. This proof is sent to the central server for aggregation and subsequently used for on-chain verification.

The evaluated proving performance results of all models in the last cluster are shown in Fig. 14. The mean proof generation time, CPU usage, and maximum RAM are around 8.15 s, 220.41%, and 1050.25 MB, respectively. These results come from proving evaluations with aggregation configurations because the federated ZKML requires one aggregated final proof for each cluster. We can see that these results align with the decentralized proving performance with aggregation settings in Fig. 12. Finally, the accuracy of the results across the three solutions is compared. Evaluation metrics are mean and median absolute errors, *mean_abs_error* and *median_abs_error*, defined by EZKL as the mean and median values of the absolute difference between the original and calibrated predictions, respectively. This calibration balances the proving accuracy and performance through quantization adjustment. As shown in Fig. 15, the decentralized ZKML solution has the highest errors with the most variable accuracy results, reflected in the large orange interquartile range. The federated model has the lowest median of the mean absolute errors, while the centralized solution has the lowest median of median absolute errors. The mean of all centralized and federated error values are out of range due to some significantly large inputs, which lead to the circuit quantization producing larger errors to guarantee proving efficiency.

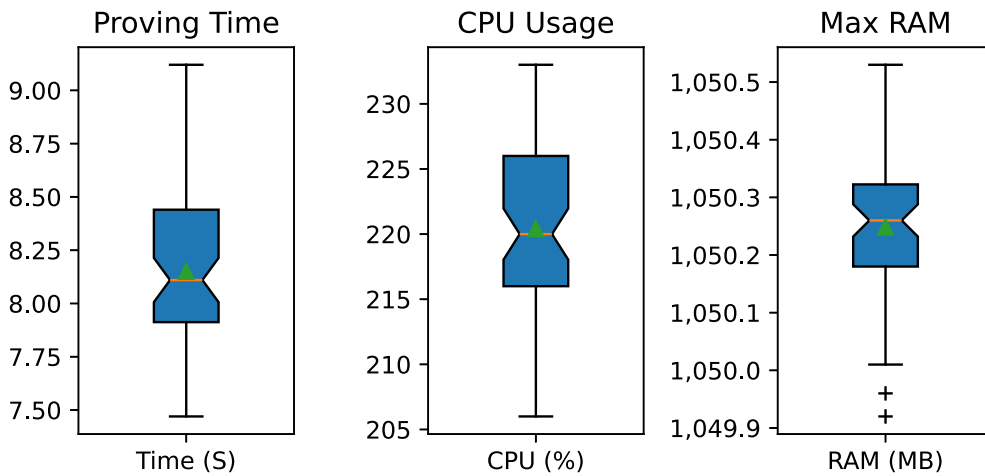


Fig. 14. Federated ZKML proving performance profiling.

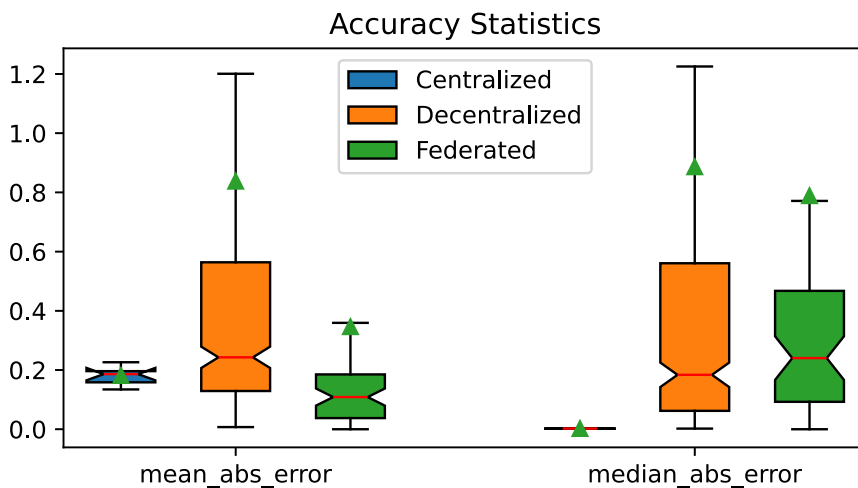


Fig. 15. Accuracy comparison of three models in terms of the mean absolute error and median absolute error.

Table 4

Proof generation performance of site 40 with the Linear model (logrows=12).

L	MSE	Input size	ONNX size	Proof size	Proof time	Verifier size	Bytecode size
96	0.113	0.768KB	9.560KB	22.442KB	0.410S	58.141KB	13.741KB
192	0.077	1.518KB	18.564KB	22.502KB	0.670S	58.141KB	13.741KB
336	0.073	2.643KB	32.064KB	22.441KB	1.090S	58.141KB	13.741KB
720	0.069	5.643KB	68.064KB	22.462KB	2.040S	58.141KB	13.741KB

This indicates that the federated ZKML has the best accuracy in terms of the mean absolute errors. Its accuracy in terms of the median absolute error metric is similar to that of the decentralized ZKML but worse than that of the centralized model.

Additional practical details of zkPET, including the proof size, verifier size, and verifier bytecode size are shown in Table 4. The results indicate that the generated proofs maintain a stable size of around 22 KB, even as input and ONNX model sizes increase linearly. This stability is due to the succinctness property of the zkSNARK proof system. More importantly, although the created verifiers (smart contracts) have an original size of 58.141 KB, their compiled bytecodes only require 13.741 KB of storage, which is small enough to be deployed on the blockchain.

7. Discussion

In this study, energy demand forecasting is employed to evaluate the proposed framework, as accurate short-term forecasting is critical in P2P-ET for determining trading volumes and pricing in near-future markets. Notably, zkPET is designed with a modular

architecture that offers both flexibility and extensibility. This design enables the framework to be adapted for additional P2P-ET tasks, such as dynamic bidding and market interaction modeling, provided that appropriate machine learning models are defined for these applications.

7.1. FL clustering

The proposed approach to clustering time-series data employs a centralized method where each household transmits its downsampled trend and seasonality components to a central server. This server then performs the clustering based on the compressed data. This approach ensures that the privacy of individual households is preserved, as only the abstracted patterns of the data are shared rather than the raw consumption data. By focusing on the trend and seasonality, the risk of sensitive information exposure is minimized while still providing enough detail for effective clustering.

Compared to the Iterative Federated Clustering Algorithm (IFCA) [55], the proposed method offers several distinct advantages for the specific context of electricity consumption data. The IFCA approach, designed for general FL environments, alternates between cluster identity estimation and model parameter optimization without explicitly addressing the unique characteristics of time-series data. The proposed method, however, is tailored to handle the high prevalence of trend and seasonality patterns in electricity consumption data. Isolating these patterns and using them for clustering ensures that the clusters formed are better representing the underlying consumption behaviors, leading to more accurate and effective federated learning.

Furthermore, the centralization of the clustering process leverages the global view of the data trends and seasonalities, enhancing the robustness of the clusters. This centralized yet privacy-preserving approach strikes a balance between the need for effective clustering and the imperative to protect user privacy, making it a suitable choice for FL applications in this domain. This careful consideration of the data's intrinsic patterns and privacy concerns underscores the rationale behind developing this novel approach.

7.2. ZKP performance

The performance of a modern ZKP system determines its practicality in integrating intelligence into the blockchain. This article focuses on the application of ZKML to energy trading within the IoT context. A current limitation lies in proving efficiency, with proof generation times still on the order of seconds. However, future work is expected to significantly reduce this overhead through hardware acceleration. GPUs, which are optimized for parallel computation, are particularly well-suited for the repetitive and parallelizable operations involved in ZKPs, such as polynomial commitments, matrix multiplications, and modular arithmetic. As such, the adoption of open-source GPU technologies is expected to substantially accelerate these computationally intensive proof generation tasks and enable faster proof aggregation.

In parallel, the development of emerging frameworks and novel cryptographic techniques continues to advance the performance of ZKPs. ModulusLabs [56] benchmarked the performance in terms of proof generation time and peak prover memory usage against model layers and the number of model parameters of six different ZKP systems: Groth16, Gemini, Winterfell, Halo2, Plonky2, and zkCNN [57]. The results show that the GKR-based zkCNN outperforms others, particularly in handling large models. GKR [58], named after its inventors Goldwasser, Kalai, and Rothblum, is an interactive proof system composed of layers, using the sumcheck protocol [59] when moving between layers. Another recent proof system, called psvCNN [60], employs an independent splitting design for improved proving parallelism, achieving 12 times faster proving time than zkCNN. Similarly, TensorPlonk [61], a new proving system under development, offers a high-performance proving process (around 226 times faster than EZKL) through optimizing matrix multiplications, non-linearities, and weight commitments. Despite these advancements and rapid evolution, the fundamental circuit-based design still makes the ZKML systems computationally expensive to provide cryptographic security. Therefore, ZKML usually requires the ML model size to be small, significantly limiting the model's practicality and accuracy.

To address this issue, an interactive fraud-proof protocol called Optimistic Machine Learning (opML) [62] has been introduced, reminiscent of the optimistic rollup systems. Compared to ZKML, opML offers cost-efficient and highly efficient ML services with minimal participation requirements.

7.3. Trust and verification

In a blockchain-based decentralized P2P-ET market, there should be no trust assumption among market participants. Trust can only be established through cryptographic verification, such as digital signatures and zkSNARKs, which ensure data integrity, authenticity, and confidentiality. In other words, a prover must demonstrate ownership of a certain piece of knowledge and generate a proof that can be easily verified by the verifier, allowing the verifier to trust the prover. Such a system has two levels of trust: trust in the prover owning the knowledge and trust in the knowledge itself.

It is worth noting that ZKML only resolves the former level of trust. In zkPET, ZKML proves that all computations in the trained ML model are executed faithfully, assuming the obtained model is correct. This proof is essential because it removes the trust assumption regarding the model provider. For example, a prosumer can use ZKML to verify the inference proof of an energy generation prediction model, ensuring that an energy AI company is not using a cheaper model that could result in a poor prediction. The latter level of trust concerns model interpretability, meaning that people tend to trust a model that can be well explained [63], which is out of the scope of this study.

8. Conclusion

This paper introduces a novel solution for privacy-preserving and intelligent peer-to-peer energy trading that harmonizes ZKML with blockchain technology. The proposed architecture strategically offloads intensive computations to off-chain without compromising data privacy and integrity through a cryptographic proof system. In particular, this study explored three types of system structures, centralized, decentralized, and federated ZKML. The first two involve training and proving the model in a centralized and decentralized manner, respectively. To implement the federated ZKML, we proposed a novel time-series clustering algorithm to resolve the data heterogeneity issues. For each classified cluster, Flower FL framework is used to train an ML model which is eventually published. The inference of this public model is then proved using private data on each local site and all proofs are aggregated for on-chain verification. The performance of the proposed solutions was carefully examined through extensive experiments with a simple linear model on real-world datasets and comparatively analyzed their performance regarding proving time, CPU usage, maximum RAM consumption, model prediction accuracy, and scalability. The results show that the federated ZKML outperforms the other two models in terms of data privacy, scalability, and prediction accuracy. These experimental results also provide valuable insights for researchers and industrial practitioners.

For future work, it is of interest to explore decentralized time-series clustering approaches to improve the communication overload and scalability of the federated ZKML. Additionally, enhancing its performance through more advanced ZKP systems dedicated to ML tasks can be a promising avenue for further research. Future work may also investigate the application of the proposed framework in alternative domains, such as supply chain and inventory management, where privacy-preserving collaborative decision-making and forecasting are equally critical.

CRedit authorship contribution statement

Caixiang Fan: Writing – original draft, Validation, Software, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization. **Amirhossein Sohrabbeig:** Writing – original draft, Software, Formal analysis. **Petr Musilek:** Writing – review & editing, Validation, Supervision, Resources, Funding acquisition.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Caixiang Fan reports financial support was provided by Sui Foundation. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work has been partially supported by Sui Foundation through their Sui Academic Research Awards. Digital Research Alliance of Canada (alliancecan.ca) and Cybera (cybera.ca) partially supported this research work through their cloud computing resources.

Data availability

Data link is included in the paper.

References

- [1] M.J.A. Baig, M.T. Iqbal, M. Jamil, J. Khan, Blockchain-based peer-to-peer energy trading system using open-source angular framework and hypertext transfer protocol, *Electronics* 12 (2) (2023) 287.
- [2] C. Fan, H. Khazaei, P. Musilek, BPET: A unified blockchain-based framework for peer-to-peer energy trading, *Futur. Internet* 16 (5) (2024) 162.
- [3] V. Veerasamy, Z. Hu, H. Qiu, S. Murshid, H.B. Gooi, H.D. Nguyen, Blockchain-enabled peer-to-peer energy trading and resilient control of microgrids, *Appl. Energy* 353 (2024) 122107.
- [4] J. Hwang, M.-i. Choi, T. Lee, S. Jeon, S. Kim, S. Park, S. Park, Energy prosumer business model using blockchain system to ensure transparency and safety, *Energy Procedia* 141 (2017) 194–198.
- [5] A. Harmanci, M. Gerstein, Quantification of private information leakage from phenotype-genotype data: linking attacks, *Nature Methods* 13 (3) (2016) 251–256.
- [6] K. Shahare, A. Mitra, D. Naware, R. Keshri, H. Suryawanshi, Performance analysis and comparison of various techniques for short-term load forecasting, *Energy Rep.* 9 (2023) 799–808.
- [7] M. Savi, F. Olivadese, Short-term energy consumption forecasting at the edge: A federated learning approach, *IEEE Access* 9 (2021) 95949–95969.
- [8] S. Alam, Deep learning applications for residential energy demand forecasting, *AI IoT Fourth Ind. Revolut. Rev.* 14 (2) (2024) 27–38.
- [9] A. Keddouda, R. Ihaddadene, A. Boukhari, A. Atia, M. Arici, N. Lebbihiat, N. Ihaddadene, Solar photovoltaic power prediction using artificial neural network and multiple regression considering ambient and operating conditions, *Energy Convers. Manage.* 288 (2023) 117186.
- [10] Y. Ledmaoui, A. El Maghraoui, M. El Aroussi, R. Saadane, A. Chebak, A. Chehri, Forecasting solar energy production: A comparative study of machine learning algorithms, *Energy Rep.* 10 (2023) 1004–1012.
- [11] A. Zafar, Y. Che, M. Faheem, M. Abubakar, S. Ali, M.S. Bhutta, Machine learning autoencoder-based parameters prediction for solar power generation systems in smart grid, *IET Smart Grid* (2024).

- [12] S. Shukla, S. Hussain, R.R. Irshad, A.A. Alattab, S. Thakur, J.G. Breslin, M.F. Hassan, S. Abimannan, S. Husain, S.M. Jameel, Network analysis in a peer-to-peer energy trading model using blockchain and machine learning, *Comput. Stand. Interfaces* 88 (2024) 103799.
- [13] C.D. Pop, M. Antal, T. Ciocara, I. Anghel, I. Salomie, Blockchain and demand response: Zero-knowledge proofs for energy transactions privacy, *Sensors* 20 (19) (2020) 5678.
- [14] T. Miyamae, F. Kozakura, M. Nakamura, S. Zhang, S. Hua, B. Pi, M. Morinaga, ZGridBC: zero-knowledge proof based scalable and private blockchain platform for smart grid, in: 2021 IEEE International Conference on Blockchain and Cryptocurrency, ICBC, IEEE, 2021, pp. 1–3.
- [15] D. Hou, J. Zhang, S. Huang, Z. Peng, J. Ma, X. Zhu, Privacy-preserving energy trading using blockchain and zero knowledge proof, in: 2022 IEEE International Conference on Blockchain (Blockchain), IEEE, 2022, pp. 412–418.
- [16] S. Goldwasser, S. Micali, C. Rackoff, The knowledge complexity of interactive proof-systems, in: *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, ACM, 2019, pp. 203–225.
- [17] A. Kate, G.M. Zaverucha, I. Goldberg, Constant-size commitments to polynomials and their applications, in: *Advances in Cryptology-ASIACRYPT 2010: 16th International Conference on the Theory and Application of Cryptology and Information Security*, Singapore, December 5-9, 2010. Proceedings 16, Springer, 2010, pp. 177–194.
- [18] J. Thaler, et al., Proofs, arguments, and zero-knowledge, *Found. Trends® Priv. Secur.* 4 (2–4) (2022) 117–660.
- [19] O. Goldreich, Y. Oren, Definitions and properties of zero-knowledge proof systems, *J. Cryptology* 7 (1) (1994) 1–32.
- [20] EZKL, What is EZKL?, 2022, URL <https://docs.ezkl.xyz/>. (Accessed 25 June 2024).
- [21] L. Grassi, D. Khovratovich, C. Rechberger, A. Roy, M. Schofnegger, Poseidon: A new hash function for {Zero – Knowledge} proof systems, in: 30th USENIX Security Symposium, USENIX Security 21, 2021, pp. 519–535.
- [22] zkconduit, Benchmarking ZKML Frameworks, 2024, URL <https://blog.ezkl.xyz/post/benchmarks/>. (Accessed 25 June 2024).
- [23] Risczero, Universal zero knowledge, 2024, URL <https://www.risczero.com/>. (Accessed 20 August 2024).
- [24] Orion, Orion, 2024, URL <https://orion.gizatech.xyz/>. (Accessed 20 August 2024).
- [25] C. Dwork, Differential privacy: A survey of results, in: *International Conference on Theory and Applications of Models of Computation*, Springer, 2008, pp. 1–19.
- [26] M.U. Hassan, M.H. Rehmani, J. Chen, DEAL: Differentially private auction for blockchain-based microgrids energy trading, *IEEE Trans. Serv. Comput.* 13 (2) (2019) 263–275.
- [27] K. Gai, Y. Wu, L. Zhu, M. Qiu, M. Shen, Privacy-preserving energy trading using consortium blockchain in smart grid, *IEEE Trans. Ind. Inform.* 15 (6) (2019) 3548–3558.
- [28] Z. Liu, C. Hu, H. Xia, T. Xiang, B. Wang, J. Chen, SPDTS: a differential privacy-based blockchain scheme for secure power data trading, *IEEE Trans. Netw. Serv. Manag.* 19 (4) (2022) 5196–5207.
- [29] C. Zhao, S. Zhao, M. Zhao, Z. Chen, C.-Z. Gao, H. Li, Y.-a. Tan, Secure multi-party computation: theory, practice and applications, *Inform. Sci.* 476 (2019) 357–372.
- [30] A. Abidin, A. Aly, S. Cleemput, M.A. Mustafa, Secure and privacy-friendly local electricity trading and billing in smart grid, 2018, arXiv preprint arXiv:1801.08354.
- [31] X. Zhou, B. Wang, Q. Guo, H. Sun, Z. Pan, N. Tian, Bidirectional privacy-preserving network-constrained peer-to-peer energy trading based on secure multiparty computation and blockchain, *IEEE Trans. Power Syst.* (2023).
- [32] A. Acar, H. Aksu, A.S. Uluagac, M. Conti, A survey on homomorphic encryption schemes: Theory and implementation, *ACM Comput. Surv. (Csur)* 51 (4) (2018) 1–35.
- [33] H. Yi, W. Lin, X. Huang, X. Cai, R. Chi, Z. Nie, Energy trading IoT system based on blockchain, *Swarm Evol. Comput.* 64 (2021) 100891.
- [34] P. Singh, M. Masud, M.S. Hossain, A. Kaur, Blockchain and homomorphic encryption-based privacy-preserving data aggregation model in smart grid, *Comput. Electr. Eng.* 93 (2021) 107209.
- [35] R. Deng, F. Luo, J. Yang, D.-W. Huang, G. Ranzi, Z.Y. Dong, Privacy preserving renewable energy trading system for residential communities, *Int. J. Electr. Power Energy Syst.* 142 (2022) 108367.
- [36] D. Stropparava, F. Rosato, L. Nespoli, V. Medici, Privacy and auditability in the local energy market of an energy community with homomorphic encryption, *Energies* 15 (15) (2022) 5386.
- [37] X. Lu, H. Guo, Trust-DETM: Distributed energy trading model based on trusted execution environment, *Mathematics* 11 (13) (2023) 2934.
- [38] M. Sabt, M. Achemlal, A. Bouabdallah, Trusted execution environment: what it is, and what it is not, in: 2015 IEEE Trustcom/BigDataSE/Ispa, vol. 1, IEEE, 2015, pp. 57–64.
- [39] Z. Xing, Z. Zhang, J. Liu, Z. Zhang, M. Li, L. Zhu, G. Russello, Zero-knowledge proof meets machine learning in verifiability: A survey, 2023, arXiv preprint arXiv:2310.14848.
- [40] T.P. Pedersen, Non-interactive and information-theoretic secure verifiable secret sharing, in: *Annual International Cryptology Conference*, Springer, 1991, pp. 129–140.
- [41] M. Petkus, Why and how zk-snark works, 2019, arXiv preprint arXiv:1906.07221.
- [42] D.J. Beutel, T. Topal, A. Mathur, X. Qiu, J. Fernandez-Marques, Y. Gao, L. Sani, K.H. Li, T. Parcollet, P.P.B. de Gusmão, et al., Flower: A friendly federated learning research framework, 2020, arXiv preprint arXiv:2007.14390.
- [43] N. Gholizadeh, P. Musilek, Federated learning with hyperparameter-based clustering for electrical load forecasting, *Internet Things* 17 (2022) 100470.
- [44] R.B. Cleveland, W.S. Cleveland, J.E. McRae, I. Terpenning, et al., STL: A seasonal-trend decomposition, *J. Off. Stat.* 6 (1) (1990) 3–73.
- [45] E. Kapustina, E. Shutov, A. Barskaya, A. Kalganova, Predicting electric energy consumption for a jerky enterprise, *Energy Power Eng.* 12 (6) (2020) 396–406.
- [46] M. Müller, Dynamic time warping, *Inf. Retr. Music. Motion* (2007) 69–84.
- [47] L. Kaufman, P.J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*, John Wiley & Sons, 2009.
- [48] D.M. Saputra, D. Saputra, L.D. Oswari, Effect of distance metrics in determining k-value in k-means clustering using elbow and silhouette method, in: *Sriwijaya International Conference on Information Technology and Its Applications, SICONIAN 2019*, Atlantis Press, 2020, pp. 341–346.
- [49] V. Buterin, *Quadratic Arithmetic Programs: from Zero to Hero*, 2016, URL <https://vitalik.eth.limo/general/2016/12/10/qap.html>. (Accessed 9 January 2025).
- [50] N. Gailly, M. Maller, A. Nitulescu, Snarkpack: Practical snark aggregation, in: *International Conference on Financial Cryptography and Data Security*, Springer, 2022, pp. 203–229.
- [51] M. Ambrona, M. Beunardeau, A.-L. Schmitt, R.R. Toledo, a P lon K: Aggregated p lon k from multi-polynomial commitment schemes, in: *International Workshop on Security*, Springer, 2023, pp. 195–213.
- [52] A. Zeng, M. Chen, L. Zhang, Q. Xu, Are transformers effective for time series forecasting? in: *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, 2023, pp. 11121–11128.
- [53] A.K. Jain, Data clustering: 50 years beyond K-means, *Pattern Recognit. Lett.* 31 (8) (2010) 651–666, <http://dx.doi.org/10.1016/j.patrec.2009.09.011>, Award winning papers from the 19th International Conference on Pattern Recognition (ICPR). URL <https://www.sciencedirect.com/science/article/pii/S0167865509002323>.
- [54] L. van der Maaten, G. Hinton, Visualizing data using t-SNE, *J. Mach. Learn. Res.* 9 (86) (2008) 2579–2605, URL <http://jmlr.org/papers/v9/vandermaaten08a.html>.

- [55] A. Ghosh, J. Chung, D. Yin, K. Ramchandran, An efficient framework for clustered federated learning, 2021, [arXiv:2006.04088](https://arxiv.org/abs/2006.04088).
- [56] ModulusLabs, The cost of intelligence: Proving machine learning inference with zero-knowledge, 2023, URL <https://medium.com/@ModulusLabs/chapter-5-the-cost-of-intelligence-da26dbf93307>. (Accessed 25 June 2024).
- [57] T. Liu, X. Xie, Y. Zhang, ZkCNN: Zero knowledge proofs for convolutional neural network predictions and accuracy, in: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, 2021, pp. 2968–2985.
- [58] S. Goldwasser, Y.T. Kalai, G.N. Rothblum, Delegating computation: interactive proofs for muggles, *J. ACM* 62 (4) (2015) 1–64.
- [59] C. Lund, L. Fortnow, H. Karloff, N. Nisan, Algebraic methods for interactive proof systems, *J. ACM* 39 (4) (1992) 859–868.
- [60] Y. Fan, B. Xu, L. Zhang, G. Tan, S. Yu, K.-C. Li, A. Zomaya, psvCNN: A zero-knowledge CNN prediction integrity verification strategy, *IEEE Trans. Cloud Comput.* (2024).
- [61] D. Kang, TensorPlonk: A “GPU” for ZKML, delivering 1,000x speedups, 2023, URL <https://medium.com/@danieldkang/tensorplonk-a-gpu-for-zkml-delivering-1-000x-speedups-d1ab0ad27e1c>. (Accessed 25 June 2024).
- [62] K. Conway, C. So, X. Yu, K. Wong, opML: Optimistic machine learning on blockchain, 2024, arXiv preprint [arXiv:2401.17555](https://arxiv.org/abs/2401.17555).
- [63] P. Schmidt, F. Biessmann, Quantifying interpretability and trust in machine learning systems, 2019, arXiv preprint [arXiv:1901.08558](https://arxiv.org/abs/1901.08558).