Frontiers in Energy Research

TYPE Original Research
PUBLISHED 26 September 2022
DOI 10.3389/fenrg.2022.992117

Privacy-preserving deep learning for electricity consumer characteristics identification

Zhixiang Zhang[1], Qian Lu[1]*, Hansong Xu[2], Guobin Xu[3], Fanyu Kong[4] and You Yu[5]

[1]College of Computer Science and Technology, Qingdao University, Qingdao, China, [2]School of Electronic Information and Electrical Engineering, Shanghai Jiaotong University, Shanghai, China, [3]Department of Computer Science, Morgan State University, Baltimore, MD, United States, [4]School of Software, Shandong University, Jinan, China, [5]Beijing Jingyi City Science and Industry Co., Ltd, Beijing, China

OPEN ACCESS

EDITED BY
Dou An,
MOE Key Laboratory for Intelligent Networks and Network Security, China

REVIEWED BY
Kuangyu Zheng,
Beihang University, China
Mingjian Jiang,
Qingdao University of Technology, China

*CORRESPONDENCE
Qian Lu,
luqian@qdu.edu.cn

SPECIALTY SECTION
This article was submitted to Smart Grids, a section of the journal Frontiers in Energy Research

RECEIVED 12 July 2022
ACCEPTED 25 July 2022
PUBLISHED 26 September 2022

CITATION
Zhang Z, Lu Q, Xu H, Xu G, Kong F and Yu Y (2022), Privacy-preserving deep learning for electricity consumer characteristics identification.
Front. Energy Res. 10:992117.
doi: 10.3389/fenrg.2022.992117

Deep learning models trained from smart meter data have proven to be effective in predicting socio-demographic characteristics of electricity consumers, which can help retailers provide personalized service to electricity customers. Traditionally, deep learning models are trained in a centralized manner to gather large amounts of data to ensure effectiveness and efficiency. However, gathering smart meter data in plaintext may raise privacy concerns since the data is privately owned by different retailers. This indicates an imminent need for privacy-preserving deep learning. This paper proposes several secure multi-party computation (MPC) protocols that enable deep learning training and inference for electricity consumer characteristics identification while keeping the retailer's raw data confidential. In our protocols, the retailers secret-share their raw data to three computational servers, which implement deep learning training and inference through lightweight replicated secret sharing techniques. We implement and benchmark multiple neural network models and optimization strategies. Comprehensive experiments are conducted on the Irish Commission for Energy Regulation (CER) dataset to verify that our MPC-based protocols have comparable performance.

KEYWORDS

machine learning, secure multi-party computation, replicated secret sharing, smart meter, characteristics identification

1 Introduction

Nowadays, smart meters are widely applied in residential households, which allow both customers and retailers to learn a large amount of accurate electricity consumption data (Wang et al., 2015; Mallapuram et al., 2017). In general, these fine-grained data are closely bound up with electricity consumption behavior of customers (Liang et al., 2019). Data analytics can extract the deeper insights from smart meter data, which can be used to enhance efficiency, save energy and improve smart grid systems. A vast amount of studies in machine learning algorithms have been applied to smart meter data, including classification, regression, clustering, and sparse coding (Chicco, 2016). Applications include non-technical loss detection (Jokar et al., 2015; Júnior et al., 2016), price strategy (Chen et al., 2016; Li et al., 2016), demand response program enrollment (Wang et al., 2016a; Chen and Liu, 2017), load forecasting (Taieb et al., 2016) and the electricity consumer characteristics identification (Beckel et al., 2014).

Understanding the relationship between electricity consumer characteristics and smart meter data benefits most participants in the electricity market. Through the estimated electricity consumer characteristics, retailers can infer consumer consumption patterns and thus improve demand response programs, provide more personalized services and promote energy efficiency. There is no doubt that this will significantly enhance the competitiveness of retailers who are proficient in this capability. On the other hand, customers will enjoy better services and save energy due to the technological advances.

In the literature, several data analysis methods are applied to extract mathematical models that enable the identification of electricity consumer characteristics from smart meter data. Generally, these methods consist of three phases: feature extraction, feature selection and classification or regression. In order to infer the socio-demographic characteristics of electricity consumers from smart meter data, Beckel et al. (2013) propose a automatic classification system called CLASS, and the characteristic prediction accuracies of this system are higher than 70%. Viegas et al. (2016) estimate the characteristics of consumers by transparent fuzzy models. Wang et al. (2016b) utilize non-negative sparse coding to extract hidden consumption patterns and implement classification using support vector machine (SVM). Zhong and Tam (2014) achieve the classification of customers by discrete Fourier transform. The majority of these works rely on manually extracting features, while the manually extracted features may not effectively model the high variability and nonlinearity of individual load profiles. To solve this problem, the emerging deep learning techniques (LeCun et al., 2015) are applied to electricity consumer characteristics identification. Wang et al. (2018) leverage convolutional neural networks (CNN) to extract the highly nonlinear features from massive load profiles, and demonstrate the effectiveness by experiments on the Irish CER dataset. Lin et al. (2021) combine CNN and long short-term memory (LSTM) to predict the household characteristics.

Training an accurate deep learning model requires a large amount of available data. However, smart meter data is privately held by different retailers. In order to solve this problem, the previous works assume that there is a server having access to the raw data of retailers so that it can provide machine learning services in a centralized manner. Note that smart meter data and socio-demographic characteristics are sensitive information for consumers. Information leakage may lead to dissatisfaction from customers and public opinion attacks from competitors. As a result, retailers may not reveal raw data to the server due to privacy concerns and potential business risks. In addition, governments are also pushing for strict regulation of data privacy. For instance, the General Data Protection Regulation (GDPR) is already in effect in the European Union.

Secure multi-party computation (MPC) (Yao, 1986; Goldreich et al., 2019) provide a solution to these privacy-preserving issues, which is an important cryptographic technique that is commonly employed in previous studies for privacy-preserving machine learning, such as SecureML (Mohassel and Zhang, 2017), ABY3 (Mohassel and Rindal, 2018), SecureNN (Wagh et al., 2019) and Falcon (Wagh et al., 2021). Secure multi-party computation enables multiple parties $P_1, \ldots, P_n$ to collectively compute a function f with their private input $x_1, \ldots, x_n$ and without revealing any information except the output. In this paper, we leverage replicated secret sharing techniques to construct MPC protocols for deep learning. The secret sharing based protocols require that all computing parties stay online during the execution process and have sufficient computing power. Consider that some retailers may not be able to meet both requirements, we assume that retailers distribute smart meter data in the form of secret shares to three servers, which provide the deep learning training and inference services. Such an outsourced computation pattern has proven to be very practical (Zhang et al., 2020; Zhang et al., 2021; Lu et al., 2022).

1.1 Our contributions

We summarize our contributions as follows.

- We design several MPC protocols that enable privacy-preserving deep learning training and inference for electricity consumer characteristics identification while keeping the retailer's raw data confidential. In our protocols, we implement two deep neural network models and multiple optimization strategies.
- To relieve the burden on retailers, we propose a system architecture that allows retailers to not have to engage in online computation. Retailers only need to upload secret shares of their smart meter data.
- To demonstrate the practicality, we implement our protocols based on the MP-SPDZ framework (Keller, 2020) and conduct a series of experiments on the Irish Commission for Energy Regulation (CER) dataset (Commission for Energy Regulation, 2012).

1.2 MPC frameworks

In recent years, MPC has evolved from theoretical research to provide practical privacy-preserving protocols for many

TABLE 1 Notations used in this paper.

Symbols	Descriptions
α	The learning rate
B	The mini-batch size
N	The number of retailers
D_n	The smart meter dataset of the nth retailer
D	The data set consisting of all D_n
l	The bit length of the arithmetic circuit
x	Lowercase bold letter denotes vector
$x^{(i)}$	The ith element of x
$[x]_M$	A arithmetic secret sharing of $x \in M$
$[x]_2$	A binary secret sharing of $\in \mathbb{Z}_2$
$[x]_B$	A vector of l binary secret sharing which encodes $x \in \mathbb{Z}_{2^l}$

machine learning tasks, such as training and evaluation of linear regression, logistic regression and neural networks (Mohassel and Zhang, 2017; Mohassel and Rindal, 2018; Wagh et al., 2019, 2021). Here, we give a brief overview on works related to our protocols. ABY3(Mohassel and Rindal, 2018) follow the same blueprint as ABY(Demmler et al., 2015) by mixing replicated secret sharing with garbled circuits. Eerikson et al. (2019) introduce an optimization that can leverage pseudo-random generator (PRG) to reduce the communication costs of input sharing in replicated secret sharing. Keller and Sun, (2021) implement purely training of neural network in MPC with 99% accuracy. Furthermore, Keller and Sun, (2021) discuss in detail how to implement various building block of secure computation with replicated secret sharing.

1.3 Road map

The rest of the paper is organized as follows: we present the problem statement in **Section 2**. In **Section 3**, we introduce basic three-party protocol. We introduce in detail how to construct the required secure computation building blocks in **Section 4**. In **Section 5**, we discuss the building blocks for deep learning. We report the experimental results in **Section 6**. Finally, we conclude this paper in **Section 7**.

2 Problem statement

2.1 Notation

We summarize the notations used in this paper in **Table 1**.

2.2 Privacy-preserving deep learning

Deep learning is broadly applied in many domains, such as language translation and image classification, often leading to breakthroughs in each domain. The model used for deep learning is a deep neural network, which consists of linear layers and nonlinear layers. Linear layers, including fully connected layers and convolutional layers, can be reduced to arithmetic operations as multiplications and vector dot products. While the activation functions required for the nonlinear layers, such as ReLU functions and max-pooling functions, can be efficiently implemented on binary circuits. Privacy-preserving deep learning is very challenging due to it involves the "mixed" evaluation of arithmetic and binary circuits. The previous works have proposed two main cryptographic approaches that can implement privacy-preserving deep learning: homomorphic encryption (Paillier, 1999; Gentry, 2009) and MPC. The (fully) homomorphic encryption is mainly used for computing linear layers in two-party (client-server model) secure neural network inference. The nonlinear layers in two-party secure neural network inference are usually implemented *via* oblivious transfer (OT) (Asharov et al., 2013) or garbled circuit (GC) (Yao, 1986), which are important cryptographic primitives of MPC. The studies on secure neural network training mainly focus on two types of three-party MPC protocols (Mohassel and Rindal, 2018; Wagh et al., 2019, 2021) for efficiency. The first have the two computing parties performing the secure neural network training by two-party additive secret sharing and the remaining party generating the materials required by the two computing parties The second utilize the three-party replicated secret sharing to accomplish the secure neural network training. In both scenarios, the participants who own the data are not directly involved in the computation, but instead distribute the raw data to the three computing parties in the form of secret shares. In this paper, we investigate how to use the three-party replicated secret sharing technique to construct privacy-preserving deep learning protocols.

2.3 System architecture

This paper targets to privacy-preserving deep learning (PPDL) for electricity consumer characteristics identification. Our system architecture is shown in **Figure 1**. At the core, there are two types of entities: the retailer and the server. Retailers are the owner of smart meter data and wish to accomplish the training of the deep neural network models. Since deep learning is a data-driven analytics approach, different retailers wish to work together to ensure the effectiveness of the models. Let N be the number of retailers. The smart meter dataset of the nth retailer is denoted by D_n $(n \in \{1, 2, \ldots, N\})$ and the dataset consisting of all D_n is denoted by D. Traditionally, this can be achieved through the Machine-Learning-as-a-Service (MLaaS) architecture, which leverages the power of the computational servers. Many major companies such as Amazon, Google, or Microsoft all provide computational services. Since smart meter data is privacy sensitive, retailers want to ensure confidentiality

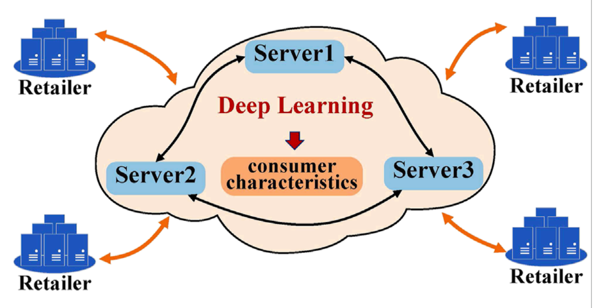

FIGURE 1
System architecture.

while enjoying the benefits of the computational servers. As a result, retailers are reluctant to supply the smart meter data in plain text, but the ciphertexts of the smart meter data are supplied instead. In addition, it may be impractical to keep all retailers online at the same time to perform the active data interactions required by the MPC protocols. Hence, our system should allow retailers to stay offline after uploading the shares of smart meter data.

In our system, three servers S_0, S_1, S_2 play the role of computing parties to ease the burden on retailers. We assume that government or other social deterrents are sufficient to make the servers strictly execute the protocol and not collude with each other. Similar to the popular security designs in recent years (Mohassel and Rindal, 2018; Wagh et al., 2021), servers use the lightweight replicated secret sharing techniques to collaboratively accomplish the secure deep neural network training. The workflow of our system is as follows. Before deep neural network training, retailers encrypt their smart meter data by splitting it into three secret shares, which can form three unique pairs. Retailers distribute each pair to a server. With shared smart meter data, servers perform the training of deep learning models by invoking various secure computation building blocks, such as dot product, secure comparison and oblivious selection. The trained model parameters are stored on the server in the form of secret shares. Retailers or other users can query the system or download the model parameters directly.

2.4 Security model

Security definition. Our protocol works under a three-party honest-majority setting in which an adversary $\mathcal{A}$ can corrupt at most one party. We assume $\mathcal{A}$ takes static corruption strategy; it decides which party to corrupt before executing the protocol. The adversary is semi-honest; it faithfully follows the protocol and attempts to learn sensitive information from protocol execution.

We use the simulation-based security definition (Canetti, 2001; Goldreich, 2009; Araki et al., 2016) for three-party computation (3PC). A 3PC protocol Π computes a functionality $f: (\{0,1\}^*)^3 \rightarrow (\{0,1\}^*)^3$. For an input tuple $\vec{x} = (x_0, x_1, x_2)$ where party P_i provides x_i, the output is $f(\vec{x}) = (f_0(\vec{x}), f_1(\vec{x}), f_2(\vec{x}))$, and P_i receives $f_i(\vec{x})$. Intuitively, Π is secure if for any corrupted party P_i, there exists a probabilistic polynomial-time simulator Sim_i who can generate a view that is indistinguishable from the one from real-world execution. Formally, let $\mathrm{View}_i^{\Pi}(1^\lambda, \vec{x})$ be the view of P_i, security is defined as follows:

Definition 1. A protocol Π securely computes a deterministic functionality f in the presence of static semi-honest adversaries if there exist a probabilistic polynomial time simulator Sim_i $(i \in \{0, 1, 2\})$ generating computationally indistinguishable view:

$$\left\{\mathrm{Sim}_i\left(1^\lambda, x_i, f_i(\vec{x})\right)\right\} \overset{c}{\equiv} \left\{\mathrm{View}_i^{\Pi}\left(1^\lambda, \vec{x}\right)\right\},$$

by only taking P_i's input x_i, output $f_i(\vec{x})$, and other allowed public information (e.g., bitlength of inputs, the size of each D_n).

Three-party decision tree evaluation. Our PPDL protocols are special cases of three-party secure computation. In our protocols, there are three servers S_0, S_1 and S_2 hold the secret shares of D. The three servers perform privacy-preserving deep learning (training and inference) using secret-sharing. Our protocols allow the servers to learn some public information during PPDL protocols. In particular, we allow S_i to learn some public information about each D (*e.g*, the number of retailers, the size of each D_n), for which we denote as $\mathcal{L}_i(D)$. Formally, let $\mathcal{F}_{\mathrm{DL}}(D) \rightarrow (\mathcal{F}_0(D), \mathcal{F}_1(D), \mathcal{F}_2(D))$ be the ideal deep learning functionality, where $\mathcal{F}_i(D)$ contains the shares of model parameters and public information $\mathcal{L}_i(D)$. A PPDL protocol Π securely computes $\mathcal{F}_{\mathrm{DL}}$ if there exist a PPT simulator Sim_i such that for any D:

$$\left\{\mathrm{Sim}_i\left(1^\lambda, x_i, \mathcal{F}_i(D)\right)\right\} \overset{c}{\equiv} \left\{\mathrm{View}_i^{\Pi}\left(1^\lambda, \mathbf{x}\right)\right\}.$$

Security in hybrid model. In this paper, we rely on necessary secure computation protocols (*e.g.*, random bit generation, domain conversion, secure comparison and dot product) to design our PPDL protocols. Since security of these secure components has already been proven secure, we will directly use their corresponding ideal functionalities in our design. This approach is known as the hybrid model (Canetti, 2001; Hazay and Lindell, 2010) and is commonly used in existing works.

3 Three-party MPC protocol

In this paper, we use replicated secret sharing techniques (Mohassel and Rindal, 2018; Eerikson et al., 2019; Keller and Sun, 2021) to construct MPC protocols for deep learning, which can be traced back to Benaloh and Leichter, (1988). We begin by introducing the basic secret sharing framework and then move on to high-level building blocks.

3.1 Secret sharing scheme

Replicated secret sharing is a variant of additive secret sharing by appending redundant shares. As we mentioned, three servers S_0, S_1 and S_2 play the role of computing parties to perform three-party MPC protocols. We denote the next and previous servers of S_i as S_{i-1}, S_{i+1}, *i.e.*, the indices are computed modulo three. The secret value x is represented as the sum of the three secret shares: $x = x_0 + x_1 + x_2 \pmod{M}$, where x_{i-1} and x_{i+1} are sent to S_i. Such a 2-out-of-3 replicated secret sharing is denoted as $[x]_M$. In this paper, we set $M = 2^l$ to utilize the properties of the ring. In general, the computation with $M = 2$ is known as binary circuits, while computing with larger moduli is called arithmetic circuits. In addition, if M is clear from context, we will omit this from the sharing notation.

3.2 Generating randomness

Throughout this paper, we require to generate randomness using pseudo-random generators (PRG). In the initialization phase, S_i and S_{i+1} share a key of PRG so that they can generate the same random number $r_{i,i+1}$. That is, each server will hold two PRG keys during the protocol. To generate a 3-out-of-3 additive secret sharing of zero, each server S_i compute $r_{i-1,i}$ and $r_{i,i+1}$ and set $r_{i,i+1} - r_{i-1,i}$ as its share. To generate a 2-out-of-3 replicated secret sharing of a random number, each server S_i compute $r_{i-1,i}$ and $r_{i,i+1}$ and set $(r_{i-1,i}, r_{i,i+1})$ as its share.

3.3 Input and open secret values

There are two types of inputting parties in our protocols. The first type of inputting parties sample and distribute secret shares from external to the servers, *e,g*, retailers. The second are computing parties, *i.e.* Servers, who need to share secret values for some building blocks. In our protocols, servers sample and distribute secret shares based on the method of Eerikson et al. (2019). If S_i wish to share a secret value x, then x_i is set to zero and x_{i-1} is generated by S_i and S_{i+1} using PRG. With x_i and x_{i-1}, S_i can compute x_{i+1} and send it to S_{i-1}.

Open secrets also have two types of situations. In order to open a secret value x to retailers or other entities, each server sends one share to the receiver, who can reconstruct x by computing $x = x_0 + x_1 + x_2 \pmod{M}$. To open a secret value x to all servers, S_i send x_{i+1} to S_{i+1}. We emphasize that the values revealed to the servers are independent of the dataset or model parameters. Hence, it does not leak sensitive information.

3.3.1 Linear operations

The additive property of the secret sharing scheme implies that linear operations can be computed locally. Let c be a public constant and $[x]$ $[y]$ be shared values. The addition of $[x]$ and $[y]$ can be computed as $[x] + [y] = [x + y] \coloneqq (x_1 + y_1, x_2 + y_2, x_3 + y_3)$. The same applies to subtraction. In addition, we define $[x \pm c]$ as $(x_1 \pm c, x_2, x_3)$ to add or subtract a shared value with a public constant. As for the scalar multiplication $c[x]$, we define as $c[x] = [cx] \coloneqq (cx_0, cx_1, cx_2)$.

3.3.2 Multiplication

The multiplication of two secret values $[x]$ and $[y]$ is shown below:

$$\begin{aligned} x \cdot y &= (x_0 + x_1 + x_2) \cdot (y_0 + y_1 + y_2) \\ &= (x_0 y_0 + x_0 y_1 + x_1 y_0) + (x_1 y_1 + x_1 y_2 + x_2 y_1) \\ &\quad + (x_2 y_2 + x_2 y_0 + x_0 y_2) \end{aligned} \tag{1}$$

We can obsever that each server can compute one summand using its own share. Let $z = xy = z_0 + z_1 + z_2$ and $z_0 = x_1 y_1 + x_1 y_2 + x_2 y_1$, $z_1 = x_2 y_2 + x_2 y_0 + x_0 y_2$, $z_2 = x_0 y_0 + x_0 y_1 + x_1 y_0$, where z_i can be locally computed by S_i. Then, servers perform the operation called *re-sharing* to hold two shares as defined. To this end, each server S_i need to sends z_i to another server. However, since z_0, z_1 and z_2 are not entirely randomized, servers need to generate a 3-out-of-3 sharing of zero to mask them. Let $(\alpha_0, \alpha_1, \alpha_2)$ be a 3-out-of-3 sharing of zero and S_i hold α_i. S_i computes $z_i' = z_i + \alpha_i$ and sends z_i' to P_{i+1} to generate 2-out-of-3 sharing $((xy)_{i-1}, (xy)_{i+1}) = (z_i', z_{i-1}')$.

4 Building blocks for secure computation

In this section, we will describe the building blocks for secure computation in the RSS setting. To the best of our knowledge, we are the first to apply these techniques to privacy-preserving deep learning for electricity consumer characteristics identification.

4.1 Multiplication of fixed-point values

Since computing with floating-point number is extremely expensive (Aliasgari et al., 2013), decimals are usually represented as fixed-point numbers in the MPC protocols, *e.g.* Catrina and Saxena, (2010). A decimal x is represented as $x = \lfloor x \cdot 2^p \rceil$, where p is a positive integer used to specify the precision. For the case of addition or subtraction, the precision of the results will not change. However, the multiplication of two fixed-point numbers doubles the precision $(x \cdot 2^p) \cdot (y \cdot 2^p) = xy \cdot 2^{2p}$, which causes the precision to accumulate until it overflows M. To address this problem, the previous works have proposed a method known as truncation. There are three ways to implement truncation.

- The easiest way is to multiply the result of each multiplication by 2^{-p}. However, this method can lead to errors with a certain probability and the absolute value of the errors is 1. For more details, we refer the readers to Mohassel and Zhang, (2017).
- The most effective way to reduce the negative impact of errors is the nearest truncation, which requires to shift the result by p bits after adding 2^{p-1} to the integer representation The nearest truncation can be instantiated by mixed-circuit computation (Dalskov et al., 2021).
- Catrina and Saxena, (2010) present a solution called probabilistic truncation that can effectively balance cost and accuracy. This method utilizes the uniformly selected random numbers. Let x be the truncated secret value and r be a random number. Servers first compute $[x + r] = [x] + [r]$ and then perform truncation. Finally, servers remove the mask r to obtain $[x]$. For instance, if $x = 0.6$, then x will round to 1 with 60% probability. In this work, we mainly use probabilistic truncation.

4.2 Dot product

The dot product is the core building block of the linear layer. Let **x** and **y** be two m-dimensional vectors. The dot product of. **x** and **y** is shown below:

$$\begin{aligned} \mathrm{x}\cdot\mathrm{y} &= \sum_{i=1}^{m} x^{(i)}\cdot y^{(i)} = \sum_{i=1}^{m}\left(x_0^{(i)}+x_1^{(i)}+x_2^{(i)}\right)\cdot\left(y_0^{(i)}+x_1^{(i)}+y_2^{(i)}\right) \\ &= \sum_{i=1}^{m}\left(x_0^{(i)}y_0^{(i)}+x_0^{(i)}y_1^{(i)}+x_1^{(i)}y_0^{(i)}\right) \\ &\quad+\sum_{i=1}^{m}\left(x_1^{(i)}y_1^{(i)}+x_1^{(i)}y_2^{(i)}+x_2^{(i)}y_1^{(i)}\right) \\ &\quad+\sum_{i=1}^{m}\left(x_2^{(i)}y_2^{(i)}+x_2^{(i)}y_0^{(i)}+x_0^{(i)}y_2^{(i)}\right) \end{aligned} \tag{2}$$

Intuitively, the dot product of x and y should be reduced to m parallel multiplications and one summation, which requires m re-sharing operations and m truncations. However, it is feasible to reduce the usage of truncation and resharing by delaying them to after the summation. Each server can first compute one of the three sums in the last term locally. Then, all servers perform re-sharing and truncation on the sums. In this way, the communication cost of one dot product is the same as a single multiplication.

4.3 Domain conversion

Recall that we use two different versions of replicated secret sharing techniques. The first is the arithmetic sharing with $M = 2^l$, which is more suitable for arithmetic operations such as addition, multiplication and dot product. The second is the binary sharing with $M = 2$, which is more suitable for binary operations and non-linear operations that need to access the individual bits directly, such as comparison. For situations that require both versions, the ideal solution is to construct efficient building blocks that allow the secret sharing of two versions to convert to each other. Especially in deep learning, domain conversion is the bridge between linear layers and nonlinear layers. For brevity, we use Bit2A and A2B to represent the conversions in two directions, respectively.

4.3.1 Random bit generation

An efficient solution of Bit2A is to leverage XOR operation, which can be defined as the function $f(x,y) = x + y - 2\cdot x\cdot y$ for $x, y \in \mathbb{F}_p$. As we can see, an XOR operation require to compute one multiplication of secret values, while using the pre-processed random bits can reduce the XOR operation to linear operations. Note that these random bits need to be secret-shared in the arithmetic circuit and no server is aware of their true value. In order to obtain such random bits, two servers sample and share a random bit respectively. Let r_0 and r_1 be the sample random bits. Then, all servers jointly compute $[r]$ by $[r] = [r_0] + [r_1] - 2\cdot[r_0]\cdot[r_1]$.

4.3.2 Bit2A

With the arithmetic sharing of random bits, Bit2A can be implemented by the idea of "daBits" (Rotaru and Wood, 2019). A daBit is a random bit that is shared in both arithmetic and Boolean circuits. Let r be a random bit and $[r]$ $[r]_2$ be available. Servers can mask a secret bit b with r and open $b\oplus r$ without leaking any information. Then, servers can remove the mask r in arithmetic circuits to obtain $[b]$.

To construct a daBit, servers need to invoke one random bit generation to obtain $[r]$. On the other hand, as introduced by Escudero et al. (2020) $[r]_2$ can be locally generated for powers of two are compatible. Observe that

$$\begin{aligned} \sum_{i=0}^{i=2}(r_i \bmod 2) \bmod 2 &= \left(\sum_{i=0}^{i=2} r_i \bmod 2^l\right) \bmod 2 \\ &= r \bmod 2 \end{aligned} \tag{3}$$

Hence, servers can locally generate $[r]_2$ by extracting the least significant bit of $[r]$.

4.3.3 A2B

A2B can be considered as a special case of bit decomposition. In this paper, we adopt the method proposed by Araki et al. (2016); Mohassel and Rindal, (2018) to perform A2B. Recall that the arithmetic sharing of x is $[x] := (x_0, x_1, x_2)$. We use $[x]_B$ to denote a vector of l binary secret sharing which encodes $x \in \mathbb{Z}_{2^l}$. We can observe that $[x_0]_B := (x_0, 0, 0)$, $[x_1]_B := (0, x_1, 0)$ and $[x_2]_B := (0, 0, x_2)$ are valid but not random binary sharings.

Then, servers can compute $[x]_B = [x_0]_B + [x_1]_B + [x_2]_B$ in the binary addition circuit.

4.4 Secure comparison

The comparison is essential for the implementation of a lot of activation functions, such as ReLU functions, max-pooling functions and approximate sigmoid functions. The comparison is defined as $b = x \overset{?}{<} y$ for $x, y \in \mathbb{Z}_{2^l}$. As introduced by Mohassel and Rindal, (2018); Keller and Sun, (2021), the most significant bit (MSB) denotes the sign of a ring element, which implies that the comparison can be reduced to the MSB extraction of the difference between the two operands. Given the secret sharings of x and y, servers first compute the difference a locally by $[a] = [y] - [x]$ and then convert the arithmetic sharing of a to its binary sharing. It remains to extract the MSB of a and convert it to the arithmetic sharing by invoking Bit2A for subsequent operations.

4.5 Oblivious selection

Oblivious Selection is an essential building block for segmentation functions. It can avoid participants learning which branch is selected. Oblivious selection can be reduced to a polynomial. Taking the 1-out-of-2 oblivious selection as an example, let x and y represent the branches and $b \in \{0, 1\}$ represent the condition. The oblivious selection can be done by $x + b \cdot (y - x)$. And so on, the oblivious selection with more branches can be implemented by polynomials with higher orders.

4.6 Division

Since arithmetic operations are performed on the ring $\mathbb{Z}_{2^l}$, we cannot compute the division directly. There are two main ways to solve this problem: sequential comparison and numerical methods. The specific method we use is the numerical method by Catrina and Saxena, (2010), which instantiates the algorithm of Goldschmidt, (1964) in MPC. This method iteratively approximates the results by multiplication. Therefore, the error of the results mainly depends on the usage of iterations.

4.7 Logarithm and exponentiation

Similar to division, logarithm and exponentiation are implemented by numerical methods (Aly and Smart, 2019). We use $\log_a x$ and x^y to represent the instances of logarithm and exponentiation, respectively, where x and y are two secret values and a is an arbitrary public base.

$\mathrm{Log}_a x$ can be reduced to $\log_a 2 \cdot \log_2 x$. Then, x is represent as $x = b \cdot 2^c$ such that $\log_2 x$ can be computed by $\log_2 x = \log_2 b + c$, where $b \in [\,0.5, 1\,)$ and $c \in \mathbb{Z}$ $\log_2 b$ can be computed by Padé approximation (Hart, 1978), which is achieved by a division of polynomials.

X^y can be reduced to $x^y = 2^{y \log_2 x}$. Computation exponentiation with base two can be done by $2^a = 2^{\lfloor a \rfloor} \cdot 2^{a - \lfloor a \rfloor}$, where the former is achieved by polynomial approximation and the latter by bit decomposition and multiplication. Let $b = \sum_{k \geq 0} b_k 2^k$ is an integer with $b_k \in \{0, 1\}$, the integer power of 2 can be computed as follows

$$2^b = 2^{b = \sum_{k \geq 0} b_k 2^k} = \prod_{k \geq 0} 2^{b_k 2^k} = \prod_{k \geq 0} \left(1 + b_k \cdot \left(2^{2^k - 1}\right)\right) \tag{4}$$

The above three operations are approximated by numerical methods, the accuracies of which depend on the number of iterations or the truncation method used for multiplication.

5 Building blocks for deep learning

In this section, we will introduce how to construct the building blocks for deep learning.

5.1 Fully connected layers

A fully connected layer is also called a dense layer, which is a linear transformation parameterized by the weight W and the bias b. Let x be the input to a fully connected layer. The output u can be computed by $u = W \cdot x + b$. The matrix multiplications are implemented by dot products. To save communication rounds, all dot products of a matrix multiplication are computed in parallel.

5.2 Convolution layers

Convolutional layers are the main layers for feature extraction. Each convolutional layer has a certain number of kernels (also known as filters). These kernels are represented as vectors so that the convolution can be performed using only dot products. Furthermore, these dot products are also computed in parallel to reduce communication rounds.

5.3 ReLU

ReLU functions (Nair and Hinton, 2010) enhance the nonlinear relationship between the layers of the neural network, which can be mathematically defined as follows

$$\mathrm{ReLU}(x) := \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

ReLU functions can be reduced to one comparison and one oblivious selection. The comparison results of forward propagation are reused in backward propagation to reduce the invocations of comparison.

5.4 Softmax

The objective of softmax functions is to present the results of multi-classification in the form of probabilities. The probability of the ith class can be computed as follows, and the classification result is the class corresponding to the maximum probability.

$$\text{Softmax}(x) = \frac{e^{x_i}}{\sum_j e^{x_j}}. \tag{6}$$

5.5 Sigmoid

Sigmoid is one of the most widely used activation functions. However, since sigmoid requires costly exponential operations, previous works usually use segmentation functions to approximate it. There are two method to approximately compute sigmoid: 3-piece approximation (Mohassel and Zhang, 2017) and 5-piece approximation (Hong et al., 2020). The two piecewise functions are shown below:

$$3-\text{piece Sigmoid}(x) \coloneqq \begin{cases} 0 & x < -0.5 \\ x + 0.5 & -0.5 \le x < 0.5 \\ 1 & x \ge 0.5 \end{cases} \tag{7}$$

$$5-\text{piece Sigmoid}(x) \coloneqq \begin{cases} 10^{-4} & x < -5 \\ 0.02776 \cdot x + 0.145 & -5 < x \le -2.5 \\ 0.17 \cdot x + 0.5 & -2.5 < x \le 2.5 \\ 0.02776 \cdot x + 0.5 & 2.5 < x \le 5 \\ 1 - 10^{-4} & x \ge 5 \end{cases} \tag{8}$$

In this way, sigmoid functions can be implemented by comparison and oblivious selection In the experiments for binary classifications, we use the 5-piece sigmoid function.

5.6 Max-pooling

Pooling layers can effectively reduce the size of the parameter matrix and thus reducing parameters in the final connection layer. Therefore, adding pooling layers can speed up the computation and prevent overfitting. In this paper, we mainly use max-pooling, the functionality of which is to return the maximum value of a small window. To reduce communication rounds, the input secret shared values are grouped in the form of a balanced tree to allow multiple comparisons to be computed in parallel.

5.7 Stochastic gradient descent (SGD)

SGD is an efficient approximation algorithm for gradually searching for a local optimum of a problem. As a widely used optimization function, SGD has proven to converge to a global minimum and is usually very fast in practice. In addition, how to securely compute SGD with MPC has been explored by a series of studies, which only involves basic arithmetic operations. As a result, we mainly focus on SGD in this paper. The workflow of SGD algorithm is as follows: the coefficients are initialized to random values or all zeros. In each iteration, a coefficient w_j is updated as

$$w_j \coloneqq w_j - \frac{\alpha}{B}\sum_{i=1}^{B}\frac{\partial l_i}{\partial w_j}. \tag{9}$$

where α is the learning rate, B is the mini-batch size and l_i is the loss regarding the ith sample in the mini-batch.

6 Case studies

In this section, we conduct a series of experiments based on the Irish CER dataset to demonstrate that our protocols not only efficiently maintain the confidentiality of the raw data, but also ensure the accuracy of the models.

6.1 Dataset description

We conduct experiments on a public dataset provided by Commission for Energy Regulation. (2012), which is the regulator for the electricity and natural gas sectors in Ireland. The CER dataset contains raw smart meter data of 4,232 residential consumers. The smart meter data is recorded at an interval of 30 min over a total of 75 weeks. In the data cleansing process, if the measurements for one of the weeks have missing data, we will delete the load profiles of this week. Besides, we limit each week starting on Friday. We select a total of 20,000 weeks of smart meter data, where the measurements of 17,000 weeks are used to train the models, and the rest are used to test the model performance.

In addition to smart meter data, the CER dataset also contains the characteristics information of the participants, which is privately collected through the questionnaire. The surveyed issues are mainly in three categories: the occupant socio-demographic information (*e.g.*, employment, social class), consumption habits (*e.g.*, the number of energy-efficient light bulbs), home appliances (*e.g.*, cooking facility type). We select

TABLE 2 The characteristics to be studied.

Question No.	Consumer Characteristic Question	Class Labels	Number
300	Age of chief income earner	Young (<35)	436
		Medium (3565)	2819
		Old (>65)	953
310	Chief income earner has retired or not	Yes	1285
		No	2947
401	Social class of chief income earner	A or B	642
		C1 or C2	1840
		D or E	1593
410	Have children or not	Yes	1229
		No	3003
450	House type	Detached or bunglow	2189
		Semi-detached or terraced	1964
453	Age of the house	Old (>30)	2151
		New(<30)	2077
460	Number of bedrooms	Very low (<3)	404
		low (=3)	1884
		High (4)	1470
		Very high (>4)	474
4704	Cooking facility type	Electrical	1272
		Not electrical	2960
4905	Energy-efficient light bulb proportion	Up to half	2041
		Three quarters or more	2191

TABLE 3 Hyperparameters of network A.

Layer	Layer Type	Hyperparameters	Activation function
FC1	Fully Connected	Input size: 7×48 Neuron number: 128	ReLU
FC2	Fully Connected	Input size: 128 Neuron number: 128	ReLU
FC3	Fully Connected	Input size: 128 Neuron number: 1	-
O1	Output	-	Sigmoid

TABLE 4 Hyperparameters of network B.

Layer	Layer Type	Hyperparameters	Activation function
C1	Convolution	Input size: 7×48 Kernel size: 3×3 Kernel number: 16	ReLU
C2	Convolution	Input size: 5×46 Kernel size: 3×3 Kernel number: 16	ReLU
P1	Max-Pooling	Window size: 2×2	-
FC1	Fully Connected	Neuron number: 32	ReLU
O1	Output	-	Softmax

nine survey questions for benchmarking, which are listed in Table 2.

6.2 Setup

We implement privacy-preserving deep learning for electricity consumer characteristics identification using the MP-SPDZ framework (Keller, 2020). The framework enables benchmarking the secure program with a series of generic MPC protocols. All experiments are run on a commodity desktop equipped with Intel (R) Core i7-11700K CPU at 3.60 GHz × 16 running Ubuntu 20.04 on VMware Workstation allocated with 32 GB memory, ignoring network latework. We set the batch size to $B = 128$ and the bit length to $l = 64$. The learning rate α is settled for 0.01. The fixed-point values are set to 16-bit precision with probabilistic truncation. We mainly use the two neural networks shown in Table 3, 4. Network A is used to train

TABLE 5 Performance of communication (MB/epoch), computation (s/epoch) and accuracy (%).

Question No.	Communication	Computation	Accuracy
300	63432	360	70.09
310	10441	24	71.83
401	63432	348	57.33
410	10441	24	74.67
450	10441	23	62.40
453	63432	353	66.53
460	63432	356	54.43
4704	10441	24	66.87
4905	10441	24	63.40

the binary-class classifiers for #310, #410, #450, #4704, #4905, while Network B is used to train the multi-class classifiers for #300, #401, #453, #460. The computation costs and accuracies reported are averaged over ten runs. The accuracies are recorded at 10 epochs.

6.3 Performance evaluation

Table 5 details the performance of the two deep neural network models we tested. Network A consists of three fully connected layers, where the first and second fully connected layers use the ReLU activation function. For the output layer of Network A, we set the sigmoid function as activation function. The computation cost required for Network A is desirable, only 24 s for each epoch. While the communication cost is 10,441 MB for each epoch. Network B contains two convolutional layers and one fully connected layer, all of which use the ReLU activation function. After the second convolution layer, we set a max-pooling layer with a window size of 2×2. For the output layer of Network B, we set the softmax function as activation function. Compared with Network A, Network B needs to invoke more secure comparisons and multiple costly building blocks, including division, logarithm and exponentiation. So, it requires more communication and computation costs. The computation cost required for Network B is around 354 s for each epoch, while the communication cost is 63,432 MB for each epoch. The communication and computation costs required are practically affordable for the resource-rich servers. In addition, the random bit generation can be performed in the preprocessing phase when servers are idle, so as to reduce the burden on servers to provide privacy-preserving deep learning services.

Now, we report the average accuracy of the survey questions. One third of the survey questions have accuracies higher than 70%, which are #300, #310 and #410. The classifiers for these three survey questions are all trained using network A. The survey question #410 has the highest accuracy of 74.67%. Only two survey questions have accuracies less than 60%, which are #401 and #460. The accuracy of the remaining survey questions is 60%～70%. In summary, the accuracy of Network A is comparable, while Network B needs to be adjusted to improve the accuracy.

7 Conclusion

We implement privacy-preserving deep learning for electricity consumer characteristics identification by lightweight replicated secret sharing techniques, which not only enable to protect the retailer's sensitive raw data but also achieve favorable performance. Our system allows retailers to stay offline after uploading the shares of smart meter data, and the burden of computation is transferred to three powerfully equipped servers. After the training of the models, retailers can enjoy the inference service provided by servers or download the model parameters directly. Future work might consider improving the accuracy of the deep neural network models.

Data availability statement

The original contributions presented in the study are included in the article/Supplementary Material, further inquiries can be directed to the corresponding author.

Author contributions

ZZ: Conceptualization, Methodology, Resources, Validation, Visualization, Writing—Original Draft, Writing—Review and Editing. QL: Funding Acquisition, Writing—Review and Editing, Project Administration. HX: Resources, Validation. GX: Validation, Visualization. FK: Software, Supervision. YY: Software, Validation, Language Modification.

Funding

This work is supported by Project funded by China Postdoctoral Science Foundation under Grant 2022T150416 and sponsored by Shanghai Pujiang program under Grant 21PJ1407000.

Conflict of interest

Author YY was employed by the company Beijing Jingyi city science and Industry Co., Ltd.

The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

References

Aliasgari, M., Blanton, M., Zhang, Y., and Steele, A. (2013). *Secure computation on floating point numbers*. (San Diego, CA, United States: computer arithmetic).

Aly, A., and Smart, N. P. (2019). "Benchmarking privacy preserving scientific operations," in International Conference on Applied Cryptography and Network Security (Springer), 509–529.

Araki, T., Furukawa, J., Lindell, Y., Nof, A., and Ohara, K. (2016). "High-throughput semi-honest secure three-party computation with an honest majority," in Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 805–817.

Asharov, G., Lindell, Y., Schneider, T., and Zohner, M. (2013). "More efficient oblivious transfer and extensions for faster secure computation," in Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security,535–548.

Beckel, C., Sadamori, L., and Santini, S. (2013). "Automatic socio-economic classification of households using electricity consumption data," in Proceedings of the fourth international conference on Future energy systems, 75–86.

Beckel, C., Sadamori, L., Staake, T., and Santini, S. (2014). Revealing household characteristics from smart meter data. *Energy* 78, 397–410. doi:10.1016/j.energy.2014.10.025

Benaloh, J., and Leichter, J. (1988). "Generalized secret sharing and monotone functions," in Conference on the Theory and Application of Cryptography (Springer), 27–35.

Canetti, R. (2001). "Universally composable security: A new paradigm for cryptographic protocols," in Proceedings 42nd IEEE Symposium on Foundations of Computer Science (IEEE), 136–145.

Catrina, O., and Saxena, A. (2010). "Secure computation with fixed-point numbers," in Financial Cryptography and Data Security, 14th International Conference, FC 2010, Tenerife, Canary Islands, January 25-28, 2010, Revised Selected Papers.

Chen, S., and Liu, C.-C. (2017). From demand response to transactive energy: State of the art. *J. Mod. Power Syst. Clean. Energy* 5, 10–19. doi:10.1007/s40565-016-0256-x

Chen, S., Love, H. A., and Liu, C.-C. (2016). Optimal opt-in residential time-of-use contract based on principal-agent theory. *IEEE Trans. Power Syst.* 31, 4415–4426. doi:10.1109/tpwrs.2016.2518020

Chicco, G. (2016). "Customer behaviour and data analytics," in 2016 International Conference and Exposition on Electrical and Power Engineering (EPE) (IEEE), 771.

Commission for Energy Regulation (CER) (2012). CER smart metering Project - electricity customer behaviour trial, 2009-2010. [Online]. Available at: https://www.ucd.ie/issda/data/commissionforenergyregulationcer/.

Dalskov, A. P. K., Escudero, D., and Keller, M. (2021). "Fantastic four: Honest-majority four-party secure computation with malicious security," in USENIX Security Symposium.

Demmler, D., Schneider, T., and Zohner, M. (2015). "Aby-a framework for efficient mixed-protocol secure two-party computation," in NDSS.

Eerikson, H., Keller, M., Orlandi, C., Pullonen, P., Puura, J., and Simkin, M. (2019). *Use your brain! arithmetic 3pc for any modulus with active security* In 5:1–5:24. ePrint Arch. doi:10.4230/LIPIcs.ITC.2020.5

Escudero, D., Ghosh, S., Keller, M., Rachuri, R., and Scholl, P. (2020). "Improved primitives for mpc over mixed arithmetic-binary circuits," in Annual International Cryptology Conference.

Gentry, C. (2009). "Fully homomorphic encryption using ideal lattices," in Proceedings of the forty-first annual ACM symposium on Theory of computing, 169–178.

Goldreich, O. (2009). *Foundations of cryptography: Volume 2, basic applications*. (Cambridge, United Kingdom: Cambridge University Press).

Goldreich, O., Micali, S., and Wigderson, A. (2019). "How to play any mental game, or a completeness theorem for protocols with honest majority," in Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali, 307–328.

Goldschmidt, R. E. (1964). *Applications of division by conPrivacy-preserving household characteristic identification with federatedvergence*. Ph.D. thesis. (Cambridge, MA, United States: Massachusetts Institute of Technology).

Hart, J. F. (1978). *Computer approximations*. (Malabar, FL, United States: Krieger Publishing Co., Inc).

Hazay, C., and Lindell, Y. (2010). *Efficient secure two-party protocols: Techniques and constructions*. (Berlin, Germany: Springer Science and Business Media).

Hong, C., Huang, Z., Lu, W.-j., Qu, H., Ma, L., Dahl, M., et al. (2020). "Privacy-preserving collaborative machine learning on genomic data using tensorflow," in Proceedings of the ACM Turing Celebration Conference-China, 39–44.

Jokar, P., Arianpoo, N., and Leung, V. C. (2015). Electricity theft detection in ami using customers' consumption patterns. *IEEE Trans. Smart Grid* 7, 216–226. doi:10.1109/tsg.2015.2425222

Júnior, L. A. P., Ramos, C. C. O., Rodrigues, D., Pereira, D. R., de Souza, A. N., da Costa, K. A. P., et al. (2016). Unsupervised non-technical losses identification through optimum-path forest. *Electr. Power Syst. Res.* 140, 413–423. doi:10.1016/j.epsr.2016.05.036

Keller, M. (2020). "Mp-spdz: A versatile framework for multi-party computation," in CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security.

Keller, M., and Sun, K. (2021). Secure quantized training for deep learning. *arXiv preprint arXiv:2107.00501*

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature* 521, 436–444. doi:10.1038/nature14539

Li, R., Wang, Z., Gu, C., Li, F., and Wu, H. (2016). A novel time-of-use tariff design based on Gaussian mixture model. *Appl. energy* 162, 1530–1536. doi:10.1016/j.apenergy.2015.02.063

Liang, H., Ma, J., Sun, R., and Du, Y. (2019). A data-driven approach for targeting residential customers for energy efficiency programs. *IEEE Trans. Smart Grid* 11, 1229–1238. doi:10.1109/tsg.2019.2933704

Lin, J., Ma, J., and Zhu, J. (2021). Privacy-preserving household characteristic identification with federated learning method. *IEEE Trans. Smart Grid* 13, 1088–1099. doi:10.1109/tsg.2021.3125677

Lu, Q., Li, S., Zhang, J., and Jiang, R. (2022). Pedr: Exploiting phase error drift range to detect full-model rogue access point attacks. *Comput. Secur.* 114, 102581. doi:10.1016/j.cose.2021.102581

Mallapuram, S., Ngwum, N., Yuan, F., Lu, C., and Yu, W. (2017). "Smart city: The state of the art, datasets, and evaluation platforms," in 2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS) (IEEE), 447–452.

Mohassel, P., and Rindal, P. (2018). "Aby3: A mixed protocol framework for machine learning," in Proceedings of the 2018 ACM SIGSAC conference on computer and communications security, 35–52.

Mohassel, P., and Zhang, Y. (2017). "Secureml: A system for scalable privacy-preserving machine learning," in 2017 IEEE symposium on security and privacy (SP) (IEEE), 19.

Nair, V., and Hinton, G. E. (2010). "Rectified linear units improve restricted Boltzmann machines vinod nair," in Proceedings of the 27th International Conference on Machine Learning (ICML-10).

Paillier, P. (1999). "Public-key cryptosystems based on composite degree residuosity classes," in International conference on the theory and applications of cryptographic techniques (Springer), 223–238.

Rotaru, D., and Wood, T. (2019). "Marbled circuits: Mixing arithmetic and boolean circuits with active security," in International Conference on Cryptology in India (Springer), 227–249.

Taieb, S. B., Huser, R., Hyndman, R. J., and Genton, M. G. (2016). Forecasting uncertainty in electricity smart meter data by boosting additive quantile regression. *IEEE Trans. Smart Grid* 7, 2448–2455. doi:10.1109/tsg.2016.25 27820

Viegas, J. L., Vieira, S. M., and Sousa, J. (2016). "Mining consumer characteristics from smart metering data through fuzzy modelling," in International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (Springer), 562–573.

Wagh, S., Gupta, D., and Chandran, N. (2019). Securenn: 3-party secure computation for neural network training. *Proc. Priv. Enhancing Technol.*, 26–49. doi:10.2478/popets-2019-0035

Wagh, S., Tople, S., Benhamouda, F., Kushilevitz, E., Mittal, P., and Rabin, T. (2021). Falcon: Honest-majority maliciously secure framework for private deep learning. *Proc. Priv. Enhancing Technol.* 1, 188–208. doi:10.2478/popets-2021-0011

Wang, Y., Chen, Q., Gan, D., Yang, J., Kirschen, D. S., and Kang, C. (2018). Deep learning-based socio-demographic information identification from smart meter data. *IEEE Trans. Smart Grid* 10, 2593–2602. doi:10.1109/tsg.2018. 2805723

Wang, Y., Chen, Q., Kang, C., and Xia, Q. (2016a). Clustering of electricity consumption behavior dynamics toward big data applications. *IEEE Trans. Smart Grid* 7, 2437–2447. doi:10.1109/tsg.2016.2548565

Wang, Y., Chen, Q., Kang, C., Xia, Q., and Luo, M. (2016b). Sparse and redundant representation-based smart meter data compression and pattern extraction. *IEEE Trans. Power Syst.* 32, 2142–2151. doi:10.1109/tpwrs.2016.2604389

Wang, Y., Chen, Q., Kang, C., Zhang, M., Wang, K., and Zhao, Y. (2015). Load profiling and its application to demand response: A review. *Tinshhua. Sci. Technol.* 20, 117–129. doi:10.1109/tst.2015.7085625

Yao, A. C.-C. (1986). "How to generate and exchange secrets," in 27th Annual Symposium on Foundations of Computer Science (sfcs 1986) (IEEE), 162–167.

Zhang, H., Gao, P., Yu, J., Lin, J., and Xiong, N. N. (2021). Machine learning on cloud with blockchain: A secure, verifiable and fair approach to outsource the linear regression. *arXiv preprint arXiv:2101.02334*

Zhang, H., Yu, J., Tian, C., Xu, G., Gao, P., and Lin, J. (2020). Practical and secure outsourcing algorithms for solving quadratic congruences in internet of things. *IEEE Internet Things J.* 7, 2968–2981. doi:10.1109/jiot.2020.29 64015

Zhong, S., and Tam, K.-S. (2014). Hierarchical classification of load profiles based on their characteristic attributes in frequency domain. *IEEE Trans. Power Syst.* 30, 2434–2441. doi:10.1109/tpwrs.2014.2362492