Privacy-Preserving Property Prediction for New Drugs with MPNN

Jiaye Xue[1], Xinying Liao[1], Ximeng Liu[1,2], and Wenzhong Guo[1,2(✉)]

[1] College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350108, China
guowenzhong@fzu.edu.cn

[2] Fujian Provincial Key Laboratory of Information Security of Network Systems, Fuzhou University, Fuzhou 350108, China

Abstract. Message passing neural network (MPNN) is one of the excellent deep learning models for drug discovery and development, drug laboratory usually outsource the MPNN model to cloud servers to save the research and development cost. However, drug-related data privacy has become a noticeable hindrance for outsourcing cooperation in drug discovery. In this paper, we propose a lightweight privacy-preserving message passing neural network framework (SecMPNN) for property prediction in new drugs. To implement SecMPNN, we design multiple protocols to perform the three stages of MPNN, namely message function, update function, and readout function. The above new-designed secure protocols enable SecMPNN to adapt to the different numbers of participating servers and different lengths of encryption requirements. Moreover, the accuracy, efficiency, and security of SecMPNN are demonstrated through comprehensive theoretical analysis and a large number of experiments. The experimental results show the communication efficiency in multiplication and comparison increases 27.78% and 58.75%, the computation error decreases to 4.64%.

Keywords: Privacy-preserving · Message passing neural network · Drug discovery

1 Introduction

Message passing neural network (MPNN) for learning molecular fingerprints has been considered to be an ideal approach to predict the properties of new drugs. Compared with other molecular prediction models, MPNN has better predictive performance and interpretability. Drug laboratory can use the detection results of MPNN to learn the properties of new drugs and make more effective improvements. However, it is quite staggering for storage capacity and the computing power requirements for training such an MPNN model for predicting the properties of new drugs. For a single drug, the MPNN model was trained with batch

J. Xiong et al. (Eds.): MobiMedia 2021, LNICST 394, pp. 446–461, 2021.
https://doi.org/10.1007/978-3-030-89814-4_32

size 20 for 3 million steps (540 epochs), and the molecular data can over 130000 samples [1]. Therefore, compared with building their own servers, the drug laboratory prefers to build the MPNN model using cloud computing technology and outsource their new drugs to cloud servers. Besides, the process from drug discovery to market costs, on average, well over $1 billion and can span 12 years or more, due to high attrition rates, rarely can one progress to market in less than ten years. Neither researcher nor laboratory wants these new drug discoveries to be revealed to some hostile adversaries. Therefore, new drug information is a valuable commercial resource for the drug laboratory, establishing an lightweight and secure privacy-preserving framework for MPNN to predict the properties of new drugs is necessary.

To address the above problems, we use the bit decomposition method to design multiple secure n-party protocols to perform the MPNN framework (SecMPNN) for predicting the properties of new drugs. The main contributions of this work are listed as follows.

- We propose SecMPNN allows drug laboratory to share new drug data securely and use a high-performance MPNN framework to give assistance to drug research. In the framework, the drug laboratory need not be anxious about their data leaks to the adversary.
- Several secure n-party sub-protocols use to finish the multiplication, comparison in the secure computation of the message function, the update function, and the readout function of MPNN without revealing the original data. Compared with traditional methods, our protocols adapt to the different numbers of participating servers and different lengths of encryption requirements.
- Through comprehensive analysis, the correctness and security of SecMPNN are proved. The experimental results show the communication efficiency in multiplication and comparison increases 27.78% and 58.75%, the computation error decreases to 4.64%.

2 Related Work

Although quantum mechanics allows us to calculate the properties of drugs in principle, the equations caused by the laws of physics are too difficult to solve accurately. As a result, scientists have developed a series of approaches to approximate quantum mechanics, with different tradeoffs between speed and accuracy, such as density functional theory (DFT) with a variety of functionals [2]. Although DFT has been widely used, its speed is still too slow to be used in large-scale systems. In order to improve the accuracy of DFT, Mohassel et al. [15] used neural networks to approximate a particularly troublesome term in DFT called the exchange-correlation potential. However, their method can not improve the efficiency of DFT and relies on a large number of special atom descriptors.

To the best of our knowledge, SecureML [5] seems to be the first privacy protection training and prediction work based on secure multiparty computing technology. However, they are inefficient and can only support very simple networks and few hidden layers. Recently, various hybrid protocol frameworks have

been proposed, like MiniONN [6] and DeepSecure [7]. However, both MiniONN and DeepSecure are computationally intensive cryptographic primitives, which are not easy to extend and can not make full use of the efficient data structure of MPNN.

3 Preliminaries

3.1 Message Passing Neural Network (MPNN)

MPNN is a sequence of connected layers that converted from an input layer to output layer. Each layer is consists of a group of neurons. Two kinds of common connected layer: fully connected layer and activation layer, are mainly used in MPNN. In [1], MPNN can be regarded as three parts: message function, update function, and readout function. We will briefly introduce these three parts as follow:

- **Message function**: The information h_v^t of each point v at time t in graph G is combined by $A_{e_{vw}}$ and its neighbor point information h_w^t, $w \in N(v)$ at time t, where $N(v)$ is the set of neighbors of point v. In detail, $A_{e_{vw}}$ represents a neural network layer as follows the pattern: $Input \rightarrow [FC \rightarrow ReLU] \times r \rightarrow FC$ where $Input$ is the attribute of edge e_{vw} and r denotes the number of repetition. This process can be express as $m_v^{t+1} = \sum_{w \in N(v)} A_{e_{vw}} h_w^t$.
- **Update function**: In this function, h_v^t update from time t to time $t+1$ according to $h_v^{t+1} = GRU(h_v^t, m_v^{t+1})$, while GRU is the gated recurrent unit. We set its specific process is as follows: $z_v^{t+1} = Sigmoid(W_{mz} m_v^{t+1} + b_{mz} + W_{hz} h_v^t + b_{hz})$, $r_v^{t+1} = Sigmoid(W_{mr} m_v^{t+1} + b_{mr} + W_{hr} h_v^t + b_{hr})$, $n_v^{t+1} = tanh(W_{mn} m_v^{t+1} + b_{mn} + r_v^{t+1} \odot (W_{hn} h_v^t + b_{hn}))$, $h_v^{t+1} = (1 - z_v^{t+1}) \odot n_v^{t+1} + z_v^{t+1} \odot h_v^t$.
- **Readout function**: After T time step, the graph information is obtained from the current stable node state and initial point state through the readout function: $R = \sum_{v \in V} Sigmoid(i(h_v^T, h_v^0)) \odot (j(h_v^T))$. In the above equation, i and j both are the neural networks, V is the point set in graph G. W and b is the weight and bias in different neural networks. $\odot$ denotes elementwise multiplication.

  Finally, prediction results R of drug properties are obtained.

3.2 Basic Definitions

We describe the basic definitions of the data format, data split and secure n-party protocol in SecMPNN.

Data Format: In SecMPNN, we convert float number to integer number by multiplying the number 10^p, and then delete the remaining decimal places, where p is the number of decimal places. It means for a floating-point number x, we compute $\bar{x} = \lfloor x \cdot 10^p \rfloor$, where $\lfloor \cdot \rfloor$ denotes the round-down operation. For simplicity,

we will omit the following overbar if there is no confusion. We use the binary complement representation of numbers to perform bit operations. The weight of the most significant bit (MSB) is a negative number of the corresponding power of 2 and the weight of other bits is a power of two. It means l-bit signed integer x can be expressed into the form $x^{(l-1)}x^{(l-2)}\dots x^{(0)}$ with $x^{(l-1)}$ being the MSB and $x = -x^{(l-1)} \cdot 2^{l-1} + \sum_{j=0}^{l-2} x^{(j)} \cdot 2^j$. The l is used for the performance benchmarks. However, it is stressed that all the protocols presented here work for any choice of l.

Data Split: We assume that there are n edge servers $S_i (i \in N_{n-1})$ participating in the work, $N_{n-1} = \{0, 1, \dots, n-1\}$. Given a number a is randomly split into n shared values as $a = \sum_{i=0}^{n-1} a_i$, where the $a_i (i \in N_{n-1})$ are called the shared values of a and will be stored in S_i, $[a] = \{a_i | i \in N_{n-1}\}$ represent the set of a_i.

Secure n-Party Protocol: All security n party agreements in this article meet the following formal definitions. Suppose $\mathcal{P}(\mathcal{I}, \mathcal{S})$ is one secure n-party protocol. Given random shared values of inputs $\mathcal{I} = \{[a], [b], \dots\}$ and n edge servers $\mathcal{S} = \{S_i | i \in N_{n-1}\}$, $\mathcal{P}$ outputs $\{f_i | i \in N_{n-1}\}$, where $\{f_i | i \in N_{n-1}\}$ are n random shared values of the computation result from $\{S_i | i \in N_{n-1}\}$ respectively. One needs to compute $f = \sum_{i=0}^{n-1} f_i$ to get the final calculation result f.

3.3 Basic Secure n-Party Protocols

The sub-protocols are introduced below which are operated among n edge serves.

Reveal: The secure n-party protocol should allow revealing the value a to all servers while covering up shared values. It can be expressed as $a = Reveal([a])$.

Secure Addition or Subtraction (SecAdd): In this protocol, servers compute $f(a, b) = a \pm b$. Since $a \pm b = \sum_{i=0}^{n-1} a_i \pm \sum_{i=0}^{n-1} b_i = \sum_{i=0}^{n-1} (a_i \pm b_i)$, it is easy to find out that $S_i (i \in N_{n-1})$ can perform the secure addition and subtraction locally without interaction with each other. Obviously, outputs satisfy $\sum_{i=0}^{n-1} f_i = a \pm b$.

Reshare: The input of this protocol is $[a]$ and output is shared value $[b]$ such that $b = a$, all shared values b_i are uniformly distributed, a_i and b_j are independent for $i, j \in N_{n-1}$. Firstly, S_i generates random $r_{i,(i+1)\%n} \in \mathbb{Z}_{2^l}$ and then $r_{i,(i+1)\%n}$ is sent from S_i to S_j. Finally, S_i computes $b_i = a_i + r_{i,(i+1)\%n} - r_{(i-1)\%n,i}$.

Secure Bit-Wise Addition Protocol (BitAdd): It is a protocol based on the *ripple-carry adder* (RCA) and carry is calculated by iterating from the least significant bit (LSB) to MSB. Given two same length of bit strings $[a^{(l-1)}]...[a^{(0)}]$ and $[b^{(l-1)}]...[b^{(0)}]$, it can be formulated as: $d^{(j)} = a^{(j)} \oplus b^{(j)} \oplus c^{(j)}$, $c^{(j+1)} = (a^{(j)}b^{(j)}) \oplus ((a^{(j)} \oplus b^{(j)})c^{(j)})$.

Secure Bit Comparison Protocol (BitCompare): It mentioned in [10] and it is trival to expand to three or more participating servers. The input is shared value of two l-bit original binary without sign bit as $[a^{(l-1)}] \dots [a^{(0)}]$ and $[b^{(l-1)}] \dots [b^{(0)}]$, the output is the shared value of result for comparison $[f]$ where $f = 1$ if and only if $[a^{(l-1)}] \dots [a^{(0)}] > [b^{(l-1)}] \dots [b^{(0)}]$.

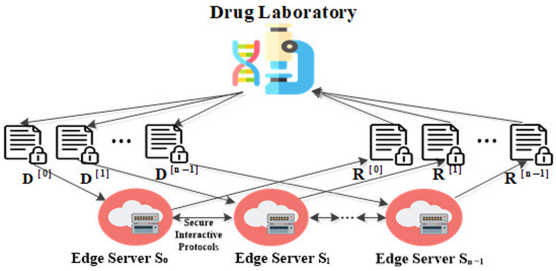


Fig. 1. System model.

4 System Architecture

4.1 System Model

There are two types of participants comprising SecMPNN shown in Fig. 1, namely the drug laboratory and the edge servers $S = \{S_i | i \in N_{n-1}\}$.

- Drug laboratory would like to use privacy-preserving property prediction for new drugs with MPNN model and is unwilling to share their new drugs data with others. In SecMPNN, drug laboratory randomly split the data D into $\{D^{[i]} | i \in N_{n-1}\}$. The encrypted data $D^{[i]} (i \in N_{n-1})$ are delivered to i^{th} edge servers.
- $S_i (i \in N_{n-1})$ are n cloud servers responsible for providing storage and computing power. In SecMPNN, They complete the calculation task of MPNN without knowing the plaintext of new drugs. The final outputs of servers $S_i (i \in N_{n-1})$ as $R^{[i]}$ are sent to drug laboratory using secure communication channels. The drug laboratory can obtain the final results by computing $R = \sum_{i=0}^{n-1} R^{[i]}$.

4.2 Attack Model

In SecMPNN, we use the model of *honest-but-curious*. In the model, $S = \{S_i | i \in N_{n-1}\}$ are all *honest-but-curious* parties, they complete each steps of protocols, and curious about the data which benefit themselves belonging to others that. In addition, we assume a simulator ζ can obtain the real view of the secure n-party protocol and generate random values. For the real view, ζ tries to generate a simulated view in polynomial time. A probabilistic polynomial algorithm can be found to distinguish the real view from the simulated view by adversary $\mathcal{A}$ is regarded as a successful attack. Also, we hypothesize that uniformly random values can be generated by edge servers, and edge servers cannot be simultaneously corrupted or collude with each other.

Algorithm 1. *SecMul*

Input: Shared values $[a]$ and $[b]$.
Output: Shared value $[f']$ such that $f' = ab$.

1: $[a'] \leftarrow Reshare([a])$.
2: $[b'] \leftarrow Reshare([b])$.
3: S_i send a'_i to servers $S_{(i+n-1)\%n}, ..., S_{(i+n-\lfloor (n-1)/2 \rfloor)\%n}$.
4: S_i send b'_i to servers $S_{(i+n-1)\%n}, ..., S_{(i+n-\lfloor n/2 \rfloor)\%n}$.
5: S_i computes $f_i \leftarrow a'_i \cdot b'_i + \sum_{j=1}^{\lfloor n/2 \rfloor} a'_i \cdot b'_{(i+j)\%n} + \sum_{j=1}^{\lfloor (n-1)/2 \rfloor} b'_i \cdot a'_{(i+j)\%n}$.
6: $[f'] \leftarrow Reshare([f])$.
7: **return** $[f']$.

5 Building Blocks: Secure *n*-Party Protocol

5.1 Secure *n*-Party Multiplication Protocol

The secure n-party multiplication protocol we designed can greatly reduce the number of rounds of communication and calculation. If we have two values a and b shared additively as $a = \sum_{i=0}^{n-1} a_i$ and $b = \sum_{j=0}^{n-1} b_j$, their product is $a \cdot b = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} a_i \cdot b_j$. The addends of the form $a_i \cdot b_i$ can be computed locally by server S_i. In order to find an addend of the form $a_i \cdot b_j$ $(i \neq j)$, the shared values a_i can be sent from S_i to S_j (or the shared values b_j from S_j to S_i). Knowing the shared values a_i and a_j, server S_j is still unable to get any information concerning a, but in order to obtain universal composability, all the shared values a_i and b_j still need to be reshared. Otherwise, the server may infer the original data from the previous information.

As illustrated in Algorithm 1, we introduce the details of secure n-party multiplication protocol under three or more participating servers.

5.2 Secure *n*-Party Comparison Protocol

The secure n-party comparison protocol is the most basic and important framework component. Suppose we have two numbers a, b to compute $f(a, b) = (a > b)$ and $f \in \{0, 1\}$, where $f = 1$ if and only if $a > b$. The completion of the secure n-party comparison protocol relies on the following security sub-protocols.

RandomBit: We now describe a protocol *RandomBit* in Algorithm 2 for securely generating shared values of a uniformly random bit. The protocol has no inputs, and the output is shared value $[r]$ of a uniformly random $r \in \{0, 1\}$. *RandomBit* protocol is based on the fact that squaring a non-zero element is a 2-to-1 mapping and given $A = \sqrt{a^2}$ one has no idea if the pre-image was a or -a. We will let the "sign" of such an a be our random bit.

RandomSolvedBits: The input of this protocol is none and output are shared values of random number $r \in \mathbb{Z}_{2^l}$ as $[r]$ and the shared values of complement $[r^{(l-1)}] \dots [r^{(0)}]$. Servers carry out *RandomBit* for l times as $[r^{(l-1)}] \dots [r^{(0)}]$ and compute $[r] = \sum_{i=0}^{l-2} 2^i \cdot [r^{(i)}] - [r^{(l-1)}] \cdot 2^{(l-1)}$ locally.

Algorithm 2. *RandomBit*

Output: Shared value $[r]$ while $r \in \{0, 1\}$.

repeat
  S_i generate random $a_i \leftarrow \mathbb{Z}_{2^l}$.
  $[a^2] = [a][a]$.
  $A^2 = \text{Reveal}([a^2])$.
until $A^2 \neq 0$.
$A \leftarrow \sqrt{A^2}$.
Select random integer $j \in N_{n-1}$.
S_i generate random $b_i \leftarrow \mathbb{Z}_{2^l}$ while $i \neq j$.
S_i compute $x_i \leftarrow b_i * A - a_i$ and send x_i to S_j while $i \neq j$.
S_j compute $b_j \leftarrow (a_j - \sum_{i \in N_{n-1}}^{i \neq j} x_i)/A$.
$[c] \leftarrow [b] + 1$.
S_i compute $y_i \leftarrow c_i \% 2$ and send y_i to S_j while $i \neq j$.
S_i compute $r_i \leftarrow (c_i - y_i)/2$ while $i \neq j$.
S_j compute $r_j \leftarrow (c_j + \sum_{i \in N_{n-1}}^{i \neq j} y_i)/2$.
return $[r]$.

Algorithm 3. *Bits*

Input: Shared value $[a]$.

Output: The shared values of complement of $[a^{(l-1)}]...[a^{(0)}]$.

$[r], [r^{(l-1)}]...[r^{(0)}] \leftarrow RandomSolvedBits()$.
$[c] \leftarrow [a] - [r]$.
$C \leftarrow Reveal([c])$.
Generate shared values of complement $[C^{(l_C)}]...[C^{(0)}]$ of $C(l_C \geq l - 1)$.
Add shared values with a prefix of zero [0] from $[r^{(l-1)}]...[r^{(0)}]$ to $[r^{(l_C)}]...[r^{(0)}]$.
$[a^{(l_C+1)}]...[a^{(0)}] \longleftarrow BitAdd([C^{(l_C)}]...[C^{(0)}], [r^{(l_C)}]...[r^{(0)}])$.
Change $[a^{(l_C+1)}]...[a^{(0)}]$ to a fixed length as $[a^{(l-1)}]...[a^{(0)}]$.
return $[a^{(l-1)}]...[a^{(0)}]$.

Bits: It is a protocol which given shared value $[a]$ securely computes the shared values of complement of $[a]$ as $[a^{(l-1)}], \ldots, [a^{(0)}]$ to realize the bit decomposition function. The input $[a]$ cover up by computing $[c] = [a] - [r]$, after writing the complement of c, use *BitAdd* protocol to add the shared values of complement of c and the shared values of complement of r to get the shared values of complement of a.

Finally, we apply the previous sub-protocol to compare two shared values. The details are shown in Algorithm 4. Note that, after getting two implementations by *Bits*, we compare the complement as the meaning of original code. Because the sign bit calculates as the highest power binary weight, it is easy to find that the result is right if a and b has the same sign and will be wrong if a and b has a different sign, we can XOR the sign bits of a and b to ensure the correctness. $\oplus$ is XOR operation for bits and it can be formulated as $[a] \oplus [b] = [a] + [b] - 2[a][b]$.

Algorithm 4. *Compare*

Input: Shared values $[a]$ and $[b]$.
Output: Shared value $[d]$ where $d = 1$ if $a > b$ and $d = 0$ otherwise.

1: $[a^{(l-1)}]...[a^{(0)}] \leftarrow Bits([a])$.
2: $[b^{(l-1)}]...[b^{(0)}] \leftarrow Bits([b])$.
3: $[c] \leftarrow BitCompare([a^{(l-1)}]...[a^{(0)}], [b^{(l-1)}]...[b^{(0)}])$.
4: $[d] \leftarrow [c] \oplus ([a^{(l-1)}] \oplus [b^{(l-1)}])$.
5: **return** $[d]$.

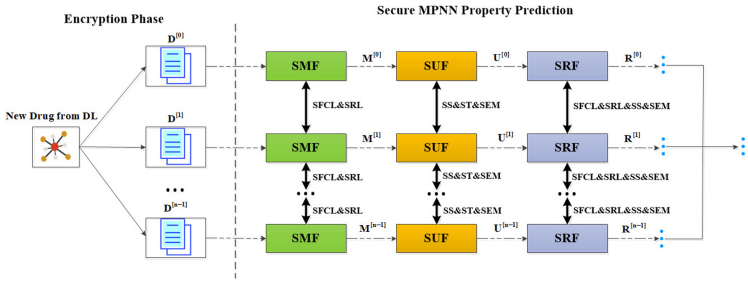


Fig. 2. Privacy-preserving MPNN for property prediction of new drugs

6 Privacy-Preserving Property Prediction of New Drugs

We provide the details of SecMPNN in this section, and discuss the feasibility of extending SecMPNN to three or more party settings. We use $x^{[i]}$ to denote the share values x, distributed in corresponding edge servers. We give an overview for SecMPNN and shown in Fig. 2. It is composed of three stages, namely secure message function (SMF), secure update function (SUF), and secure readout function (SRF). In SecMPNN, new drug data will be firstly split into random shared values and uploaded to the edge servers. Then, the following secure functions are performed securely.

6.1 Secure Message Function

By using SMF to extract new drug node features, it can be ensured that no plaintext information is leaked to the edge server. The input of SMF is the shared feature of chemical bond in drug molecules with a fixed size as $\{D^{[i]}|i \in N_{n-1}\}$, which are uploaded by the drug laboratory. SMF consists of two kinds of layers: secure fully connected layer (SFCL) and secure ReLU layer (SRL). In SFCL, specifically, for the j^{th} hidden neuron in this layer, $S_i(i \in N_{n-1})$ will compute one component of the activation: $y_j^{[i]} = \sum_k w_{j,k} x_k^{[i]} + b_j^{[i]}$. We use x_k for the activation of the k^{th} neuron in the previous layer. In addition, we use $w_{j,k}$ to denote the weights and b_j to denote the bias of the j^{th} neuron in the current layer. In SRL, we utilize $s = Compare([0], [x])$ protocol to compare input $[x]$ and 0 to determine the sign of x. If $s = 0$, the result of SRL is $[x]$; otherwise, the

result is [0]. There are 3 SFCLs and 2 SRLs deployed to implement SMF. After SMF, edge servers output $M = \{M^{[i]} | i \in N_{n-1}\}$ and send to SUF.

6.2 Secure Update Function

One part comprises the SUF, namely the secure gated recurrent unit (SGRU). The goal of the secure update function is to update each node feature without leaking any new drug feature information. Given the node feature output M by SMF, SUF set the secure update gates z and secure reset gates r by secure logistic $Sigmoid$ function (SS), use secure logistic $tanh$ function (ST) and secure element-wise multiplication (SEM) compute two gates hidden status. Note that, SS and ST can be approximated with piecewise polynomials:

$$f(x) = \begin{cases} P_0(x), & x_0 \leq x \leq x_1 \\ P_1(x), & x_1 \leq x \leq x_2 \\ & \dots \\ P_{k-1}(x), & x_{k-1} \leq x \leq x_k \end{cases} \tag{1}$$

Where $P(x)$ is an p-degree polynomial, $P(x) = a_0 + a_1 x + \ldots + a_p x^p$, $a_i (i \in Np)$ are the public weights. The higher the degree of the polynomial, the better the approximation effect that can be obtained. In order to better protect the privacy of the elements x, we use $SecMul$ protocol to compute x^p and figure out that the polynomials can be solved securely by applying the $Compare$ protocol. For SEM, we only need to use $SecMul$ protocol to compute the corresponding element securely and no other complex operations are required. After SUF, edge servers output $U = \{U^{[i]} | i \in N_{n-1}\}$ and send to SRF.

6.3 Secure Readout Function

The destination of SRF is to complete extraction of property prediction from the feature graph. In SRF, two fully connected networks use SFCL and SRL, they can execute as securely as the SMF mentioned above. We utilize SS and SEM to connect the results of two fully connected networks, they can execute as securely as SUF mentioned above. There are 6 SFCLs, 3 SRLs, 1 SS and 1 SEM deployed to implement SMF. Finally, the shared values of output R will send to the drug laboratory securely. After SRF, edge servers output $R = \{R^{[i]} | i \in N_{n-1}\}$ and send to drug laboratory.

7 Theoretical Analysis

7.1 Correctness

Firstly, in the SMF, the SFCL actually performs linear dot product. For the SRL, the relationship between data x and 0 is determined by $Compare$ protocol. If x is greater than 0, the shared value of x is retained; otherwise, the shared value of

x of all servers will become the shared value of 0. Secondly, in the SUF, SS and ST are fitted by additive function under the condition of ensuring the accuracy, and there are dot product operation of SFCL and SEM which can express as $\sum_{i=0}^{n-1} y_i = \sum_{i=0}^{n-1} Wx_i = W\sum_{i=0}^{n-1} x_i = Wx = y$, these operations also meet the additive conditions. Finally, in the SRF, SS and SEM also exist in this function.

7.2 Security

Through the general composability framework [11], We prove the security of our protocol. In our *honest-but-curious* model, adversaries were allowed to destroy up to one of the n servers. To prove that the protocol is secure enough to prove that, given its input and output, the corrupted party's viewpoint is simulatable. Specifically, we use the following definitions and lemmas.

Definition 1. *We say that if there is a probabilistic polynomial time simulator ζ protocol is secure, the simulator can generate a view of opponent $\mathcal{A}$ in the real world, and the view is computationally indistinguishable from its real view.*

Lemma 1. *A protocol is perfectly simulatable if all its sub-protocols are perfectly simulatable.*

Lemma 2. *If a random element r is uniformly distributed on $\mathbb{Z}_n$ and independent from any variable $x \in \mathbb{Z}_n$, then $r \pm x$ is also uniformly random and independent from x.*

Lemma 3. *The protocol Reshare, BitAdd, Reveal, BitCompare is secure in the honest-but-curious model.*

We advise the reader to [9] and [8] for the proofs of Lemma 1 and Lemma 2, [10] and [8] for the proof of Lemma 3. Since performing locally can be perfectly simulated, we mainly prove the security for the sub-protocols in our framework that need interactions among servers $S_i(i \in N_{n-1})$ in the following.

Theorem 1. *The protocol SecMul is secure in the honest-but-curious model.*

Proof. For $S_i(i \in N_{n-1})$, the view in the protocol execution will be $\mathcal{V}_i = (a'_i, \ldots, a'_{(i+\lfloor (n-1)/2 \rfloor)\%n}, b'_i, \ldots, b'_{(i+\lfloor n/2 \rfloor)\%n})$. These values are obtained through the *Reshare* protocol. Therefore, $\mathcal{V}_i$ is simulatable by the simulator ζ. Besides, the output of S_i will be $\mathcal{O}_i = (f_i)$, where $f_i = a'_i \cdot b'_i + \sum_{j=1}^{\lfloor n/2 \rfloor} a'_i \cdot b'_{(i+j)\%n} + \sum_{j=1}^{\lfloor (n-1)/2 \rfloor} b'_i \cdot a'_{(i+j)\%n}$. Since the operations are performed locally by S_i, $\mathcal{O}_i$ is also simulatable by the simulator ζ.

Theorem 2. *The protocol RandomBit, RandomSolvedBits, Bits, Compare are secure in the honest-but-curious model.*

Proof. For $S_i(i \in N_{n-1}, i \neq j)$, the view in the protocol execution will be $\mathcal{V}_i = (A, a_i, x_i, b_i, c_i, y_i)$. For S_j, the view will be $\mathcal{V}_j = (A, a_j, [x], b_j, c_j, y_i)$. A and $[a]$ are obtained through the *Reshare* protocol and *Reveal* protocol. $[b], [c], [y]$ are

random number. Besides, the output of S_i will be $\mathcal{O}_i = (r_i)$ where $r_i = (c_i - y_i)/2$, the output of S_j will be $\mathcal{O}_j = (r_j)$ where $r_j = (c_j + \sum_{i \in N_{n-1}}^{i \neq j} y_i)/2$. According to Lemma 2, these values are uniformly random. The proof of security about $RandomSolvedBits$, $Bits$ and $Compare$ are similar as above. It is trivial to see that they are secure in the *honest-but-curious* model.

Table 1. Comparison of the protocol complexities (Here, l is the bit-width, and $m = log_2 l$.)

Approach	$SecMul$		$Compare$		Number of servers
	Rounds	Comm(bits)	Rounds	Comm(bits)	
[8]	1	$15l$	$m+3$	$5m^2 + 12(l+1)m$	3
[3]	1	$3lm$	44	$205l + 188lm$	3
[4]	1	$3lm$	15	$279l + 5$	3
Ours	1	$(3n + \lfloor n/2 \rfloor + \lfloor (n-1)/2 \rfloor) * l$	$l+1$	$66nl$	n

Theorem 3. *The interactive protocol of secure message function, secure update function, and secure readout function in SecMPNN are secure in the semi-honest model.*

Proof. $\mathcal{A}$ eavesdrops on the transmission channels among the n edge servers and records the messages about the interactive protocols inputs into an input tape $tape_{in}$ and outputs into an output tape $tape_{out}$. According to the definitions of the interactive protocols, $\mathcal{A}$ have $tape_{in} = \mathcal{V}_{SecAdd} \cup \mathcal{V}_{SecMul} \cup \mathcal{V}_{Compare}$ and $tape_{out} = \mathcal{O}_{SecAdd} \cup \mathcal{O}_{SecMul} \cup \mathcal{O}_{Compare}$. Here, the elements belonging to the same sub-protocol are pushed into the same view. Based on Lemma 1, $tape_{in}$ and $tape_{out}$ are simulatable. It is trival to deduce SecMPNN are secure in the *honest-but-curious* model.

7.3 Efficiency

As shown in Table 1, the total communications of $SecMul$ and $Compare$ in our framework are far less than those in the traditional work. While $l = 32$ and $n = 3$, the communication efficiency in multiplication and comparison increases 27.78% and 58.75%.

8 Experiment

In this section, we first present the experimental results of our framework about the secure n-party comparison protocol. Then, we evaluate the performance and security of SecMPNN. The experiments are conducted through QM9 data, which can be obtained publicly by [12]. In the experiment, we take the SMILES [13]

string encoding of each drug as input, which is then converted into fingerprints using RDKit [14]. After that, we encrypt the fingerprints, and then send them to multiple edge servers. By adjusting the number of servers and the length of encrypted bits, we test the performance of SecMPNN under different conditions. Each server is equipped with an Intel(R) Core(TM) i5-9500 CPU @3.00 GHz and 8.00 GB of RAM.

8.1 Performance Evaluation of the Sub-protocols in SecMPNN

In our framework, since our secure n-party comparison protocol is based on bit decomposition, when evaluating the performance of the comparison protocol, we use the bit-width$l$ of the input data and the number of servers$n$ as variables, and test under different server numbers and bit widths. As shown in Fig. 4 (1)–(2), the runtime and the communication overhead both go up in the bit-width l and the number of servers n. However, they are in a matter of "milliseconds" and "Kilobyte".

8.2 Performance Evaluation of SecMPNN

We compared SecMPNN with existing other frameworks [16–18,20,21]. In [16], it proposed a cooperative learning framework and applied it to a specific region between two networks. It can extract features from the source network to other specific networks, but the optimization of the source network is ignored. In [17], it designed a secure model to encrypted data, however, it is difficult to apply it in practice because of its large communication and time overhead. In [18], it proposes a private protocol that can reduce the transmission in multiple servers. In [20,21], they achieved encryption with lightweight data, but the assumed trusted third party is difficult to find in practice. We put a summary of the comparison in Table 2.

Table 2. Summary of Comparative analysis

Function	[16]	[17]	[18]	[20]	[21]	Ours
F_1	NN	RF	LM	NN	LM	NN
F_2	×	HE	MPC	MPC	MPC	MPC
F_3	×	√	√	×	×	√
F_4	×	√	√	×	×	√
F_5	√	×	√	√	√	√

Notes. F_1: machine learning model: neural network (NN) or random forest (RF) or linear model (LM). F_2: Encrypted methods: homomorphic encryption (HE) or multi-party computation (MPC). F_3: without trusted third party. F_4: secure bit length. F_5: supporting lightweight.

Efficiency. Firstly, we test the cost of the drug laboratory for new drugs encryption. We can see from Fig. 4 (3) that the time required for encryption shows a linear increase with the increase in the number of servers and the increase in bit width. However, the time cost of the encryption and decryption stages in our framework is almost negligible. Secondly, we test the performance of our privacy-preserving MPNN property prediction framework that is performed by the different number of participating servers. Figure 4 (4)–(6) shows communication overhead of different functions for processing one instance with the networks as Fig. 3. The overall time of SecMPNN is mainly determined by the network condition. In the test with an encryption length of $l = 64$, each servers only consume about 33 Gigabyte of communication overhead to obtain the property prediction. This is mainly because we don't rely on any heavy cryptographic primitives.

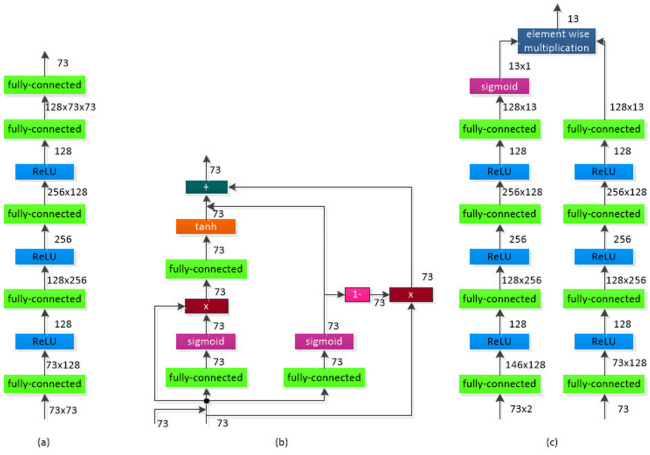
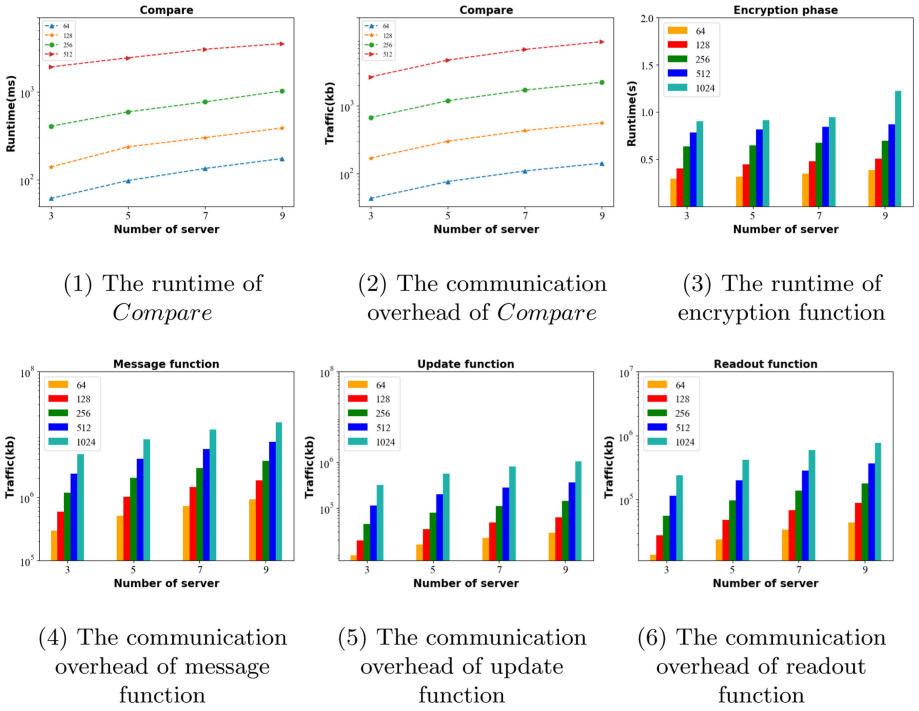


Fig. 3. MPNN architectures: (a) message function, (b) update function, and (c) readout function

Accuracy. In our SecMPNN, after the prediction of 50 drug formulas in the validation set, the average calculation error between the predicted value and the standard value is 4.64% and the accuracy is 95.36%. The calculation error mainly comes from the truncation error caused by the same data format as the integer, and the approximated error caused by approximating with piecewise polynomials of the nonlinear activation layer. We compare with the similar framework proposed in [19]. In [19], the accuracy of privacy-preserving and verifiable federated learning framework is 87.74%. It can be found that our SecMPNN framework can predict the properties of new drugs with high accuracy.



(1) The runtime of *Compare*

(2) The communication overhead of *Compare*

(3) The runtime of encryption function

(4) The communication overhead of message function

(5) The communication overhead of update function

(6) The communication overhead of readout function

Fig. 4. The performances with different number of participating servers and different length of encryption requirements.

9 Conclusions

In this paper, we proposed a novel lightweight framework for new drugs prediction as SecMPNN. The property prediction could be performed among edge servers securely. We first divide the drug data into shared values randomly, and then send them to the edge server. Using the bit decomposition method, a series of secure n-party protocols corresponding to different functions of MPNN are designed. Through empirical experiments and theoretical analysis, the security, effectiveness and accuracy of the framework are verified. In the future, we will further study more complex protocols to deal with more complex practical problems.

Acknowledgment. This work is supported by the National Natural Science Foundation of China (Grant No. U1705262, No. 62072109, No. U1804263), and the Natural Science Foundation of Fujian Province (Grant No. 2018J07005).

References

1. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: International Conference on Machine Learning, pp. 1263–1272. PMLR (July 2017)
2. Beck, A.D.: Density-functional thermochemistry. III. The role of exact exchange. J. Chem. Phys. **98**(7), 5648–5656 (1993)
3. Damgård, I., Fitzi, M., Kiltz, E., Nielsen, J.B., Toft, T.: Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 285–304. Springer, Heidelberg (2006). https://doi.org/10.1007/11681878_15
4. Nishide, T., Ohta, K.: Multiparty computation for interval, equality, and comparison without bit-decomposition protocol. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 343–360. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-71677-8_23
5. Mohassel, P., Zhang, Y.: SecureML: a system for scalable privacy-preserving machine learning. In: 2017 IEEE Symposium on Security and Privacy (SP), pp. 19–38. IEEE (May 2017)
6. Liu, J., Juuti, M., Lu, Y., Asokan, N.: Oblivious neural network predictions via miniONN transformations. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 619–631 (October 2017)
7. Rouhani, B.D., Riazi, M.S., Koushanfar, F.: DeepSecure: scalable provably-secure deep learning. In: Proceedings of the 55th Annual Design Automation Conference, pp. 1–6 (June 2018)
8. Bogdanov, D., Niitsoo, M., Toft, T., Willemson, J.: High-performance secure multiparty computation for data mining applications. Int. J. Inf. Secur. **11**(6), 403–418 (2012)
9. Bogdanov, D., Laur, S., Willemson, J.: Sharemind: a framework for fast privacy-preserving computations. In: Jajodia, S., Lopez, J. (eds.) ESORICS 2008. LNCS, vol. 5283, pp. 192–206. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-88313-5_13
10. Cramer, R., Damgård, I.B.: Secure Multiparty Computation. Cambridge University Press, Cambridge (2015)
11. Canetti, R.: Universally composable security: a new paradigm for cryptographic protocols. In: Proceedings 42nd IEEE Symposium on Foundations of Computer Science, pp. 136–145. IEEE (October 2001)
12. Ramakrishnan, R., Dral, P.O., Rupp, M., Von Lilienfeld, O.A.: Quantum chemistry structures and properties of 134 kilo molecules. Sci. Data **1**(1), 1–7 (2014)
13. Weininger, D.: SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. J. Chem. Inf. Comput. Sci. **28**(1), 31–36 (1988)
14. RDKit: Open-source cheminformatics. www.rdkit.org. Accessed 11 Apr 2013
15. Hu, L., Wang, X., Wong, L., Chen, G.: Combined first-principles calculation and neural-network correction approach for heat of formation. J. Chem. Phys. **119**(22), 11501–11507 (2003)
16. Wu, S., Zhong, J., Cao, W., Li, R., Yu, Z., Wong, H.S.: Improving domain-specific classification by collaborative learning with adaptation networks. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, no. 01, pp. 5450–5457 (July 2019)

17. Ma, Z., Ma, J., Miao, Y., Liu, X.: Privacy-preserving and high-accurate outsourced disease predictor on random forest. Inf. Sci. **496**, 225–241 (2019)
18. Zheng, W., Popa, R.A., Gonzalez, J.E., Stoica, I.: Helen: maliciously secure coopetitive learning for linear models. In: 2019 IEEE Symposium on Security and Privacy (SP), pp. 724–738. IEEE (May 2019)
19. Xu, G., Li, H., Liu, S., Yang, K., Lin, X.: Verifynet: secure and verifiable federated learning. IEEE Trans. Inf. Forensics Secur. **15**, 911–926 (2019)
20. Xia, Z., Gu, Q., Xiong, L., Zhou, W., Weng, J.: Privacy-preserving image retrieval based on additive secret sharing. arXiv preprint arXiv:2009.06893 (2020)
21. Xia, Z., Gu, Q., Zhou, W., Xiong, L., Weng, J.: Secure computation on additive shares. arXiv preprint arXiv:2009.13153 (2020)