Wang *et al. Journal of Cloud Computing: Advances, Systems and Applications* (2020) 9:36
https://doi.org/10.1186/s13677-020-00186-7

Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH Open Access

A novel privacy-preserving speech recognition framework using bidirectional LSTM

Qingren Wang[1], Chuankai Feng[1], Yan Xu[1*], Hong Zhong[1] and Victor S. Sheng[2]

Abstract

Utilizing speech as the transmission medium in Internet of things (IoTs) is an effective way to reduce latency while improving the efficiency of human-machine interaction. In the field of speech recognition, Recurrent Neural Network (RNN) has significant advantages to achieve accuracy improvement on speech recognition. However, some of RNN-based intelligence speech recognition applications are insufficient in the privacy-preserving of speech data, and others with privacy-preserving are time-consuming, especially about model training and speech recognition. Therefore, in this paper we propose a novel **P**rivacy-preserving **S**peech **R**ecognition framework using **B**idirectional **L**ong short-term memory neural network, namely PSRBL. On the one hand, PSRBL designs new functions to construct security activation functions by combing with an additive secret sharing protocol, namely a secure piecewise-linear *Sigmoid* and a secure piecewise-linear *Tanh* respectively, to achieve privacy-preserving of speech data during speech recognition process running on edge servers. On the other hand, in order to reduce the time spent on both the training and the recognition of the speech model while keeping high accuracy during speech recognition process, PSRBL first utilizes secure activation functions to refit original activation functions in the bidirectional **L**ong **S**hort-**T**erm **M**emory neural network (LSTM), and then makes full use of the left and the right context information of speech data by employing bidirectional LSTM. Experiments conducted on the speech dataset TIMIT show that our framework PSRBL performs well. Specifically compared with the state-of-the-art ones, PSRBL significantly reduces the time consumption on both the training and the recognition of the speech model under the premise that PSRBL and the comparisons are consistent in the privacy-preserving of speech data.

Keywords: Bidirectional LSTM, Privacy-preserving, Speech recognition, Edge-cloud computing, Internet of things

Introduction

Utilizing speech as the transmission medium in Internet of things (IoTs) is an effective way to reduce latency while improving the efficiency of human-machine interactions. For example, the Siri from the Apple and the Cortana from the Microsoft obtain instructions through speech recognition and return the most matched results to users, which greatly improves users' work effectiveness. Due to the advantages of speech, many applications of "IoTs + speech" in our daily life have been promoted and developed, such as smart home [1] and self-driving vehicles [2]. Studies have demonstrated that speech recognition applications based on Recurrent Neural Network (RNN) perform well in terms of improving accuracy of speech recognition [3]. However, these RNN-based speech recognition applications deployed on a centralized cloud challenge the transmission performance of devices used in IoTs and the computing performance of the centralized cloud, since they are computation-intensive and require high capacity memory. The global edge computing market size is anticipated to reach USD 43.4 billion by 2027, exhibiting a CAGR of 37.4% over the forecast period, according to

*Correspondence: xuyan@ahu.edu.cn
[1]Key Laboratory of Intelligent Computing and Signal Processing of Ministry of Education, School of Computer Science and Technology, Anhui University, 111 Jiulong Road, Hefei, China
Full list of author information is available at the end of the article

a new report by Grand View Research, Inc[1]. In addition, the data traffic of these explosively increasing terminal devices is transmitted to the cloud for processing, which will eventually exceed the cloud's computing and storage capabilities. Thus, most of RNN-based speech recognition applications are deployed on edge servers to alleviate challenges derived from computing-intensiveness and insufficient storage capabilities of the clouds [4, 5]. However, in the era of big data with the explosive growth of data volume, deploying a speech recognition application on edge servers is not an effective solution since edge computing suffers from capacity limitations. Hence, the edge-cloud computing paradigm offers a tradeoff between speech recognition applications' requirements for computing resources and low latency, and improves the usage efficiency of the IoT devices [6].

It is no doubt that there is a lack of security technologies and privacy protection mechanisms for smart speech facilitated by the edge-cloud computing paradigm [7, 8]. That is, the speech data involving private information could be maliciously collected, transmitted and analyzed during the process of speech recognition, which enables criminals to grasp users' daily behaviors and health statuses, etc. [9]. Secret Sharing [10], homomorphic encryption [11] and differential privacy [12] are three conventional algorithms for data privacy-preserving. The homomorphic encryption-based privacy-preserving methods are computation-intensive and require high capacity memory, which reduces their commercial practicability. In contrast, for privacy-preserving methods based on differential privacy, the added noise usually results in poor availability, which increases the occurrence probability of errors during the process of speech recognition. Secure Multi-Party Computing (SMC) [13] is an excellent method that could be well integrated with edge servers to improve the performance of privacy-preserving while ensuring high data availability. In addition, for speech recognition, Long Short-Term Memory neural network (LSTM) is a well-known method with high accuracy and practicability. Therefore, Ma et al. [14] proposed an outsourced privacy-preserving speech recognition framework (namely OPSR) based on advantages of SMC and LSTM.

Although OPSR greatly improves the commercial practicability of speech data and the accuracy of speech recognition while realizing the privacy-preserving of speech data, it still can be updated in terms of semantic dependence and time consumption. On the one hand, since studies [15, 16] showed that the accuracy of speech recognition could be improved by extracting the bi-directional semantic dependence of speech, the right context of speech sequence data should not be neglected. For example, due to the variety of noises coming from natural environments, the recording devices may lose some information of speech data, so that the thoughts of speakers cannot be fully expressed. Utilizing both the left and the right context information of the missing speech data is a significant way to infer the missing speech information, so as to achieve the accuracy improvement of speech data prediction. On the other hand, the activation functions used in OPSR require multiple iterations during the process of speech recognition. Intuitively, more iterations consume more time. Hence, it is necessary to seek another suitable ways instead of multiple iterations.

Therefore, this paper designs a novel **P**rivacy-preserving **S**peech **R**ecognition framework using the **Bi**directional **L**ong short-term memory neural network, namely PSRBL. Based on SMC and **Bi**-directional LSTM (BiLSTM in short), PSRBL first divides the original speech data into two encrypted parts randomly by combining with an additive secret sharing protocol. Then it utilizes new piecewise-linear functions proposed in this paper to reconstruct the original activation functions in BiLSTM. After that, PSRBL inputs the encrypted speech data into BiLSTM. Finally, PSRBL achieves the privacy-preserving of speech data based on collaborations between two independent edge servers during the training and the recognition of the speech recognition model. The experimental results demonstrate that our PSRBL significantly reduces the time consumption in terms of the training and the recognition of models, and also improves the response speed while preserving the privacy information of the speech data on the edge servers. The main contributions of this paper are summarized as follows.

(1) We propose a novel speech recognition framework with privacy-preserving based on SMC and BiLSTM, namely PSRBL. The framework PSRBL can achieve the privacy-preserving of speech data by running SMC method on two independent edge servers.
(2) We design new piecewise-linear functions to refit the original activation functions (i.e., *Sigmoid* and *Tanh*) in BiLSTM. Compared with the state-of-the-art OPSR framework, our PSRBL greatly reduces the time consumption in terms of model training and speech recognition while ensuring the consistency of training error.

The remainder of this paper is organized as follows. "PSRBL framework" presents our PSRBL framework. "Experiments" reports our experimental results. In "Related work", we overview recent work focusing on edge-cloud computing and speech recognition with privacy-preserving, and then we conclude the paper in "Conclusion" sections.

[1] https://www.grandviewresearch.com/press-release/global-edge-computing-market

PSRBL framework

In this section, we will first present the architecture of PSRBL. Then the secure forward propagation and the secure backward propagation of BiLSTM (the core of PSRBL) will be introduced respectively. Finally, we will analyze the correctness and the security capability of BiLSTM.

The architecture of PSRBL

In this paper, we achieve improvement and optimization based on OPSR and propose the framework PSRBL. The architecture of PSRBL is shown in Fig. 1 (which will be described in the next paragraphs). As can be seen in Fig. 1, our PSRBL consists of seven participants, which are end-users, the smart audio devices that end-users use, two edge servers, the trusted third party, the smart IoT devices and the service providers.

Here, U denotes an end-user. AD denotes the smart audio device that an end-user uses, and it can record and preprocess the voice data of the end-user as well as receive the results of speech recognition. Besides, AD randomly divides the feature vector of the preprocessed data into two participants, denoted as A_1 and A_2 respectively, and sends them to the two edge servers, which are denoted as S_1 and S_2 respectively. Note that for each edge server there is a training system deployed. S_1 and S_2 employ BiLSTM for speech recognition training and work together to obtain the complete output, denoted as f. T denotes the trusted third party who supports the collaborative computing of S_1 and S_2. I denotes the smart IoT devices receiving the original output results (which are denoted as f_1 and f_2 respectively) from BiLSTM. I performs the summation verification on f_1 and f_2 to get f, so as to obtain the speech recognition results that end-users require. SP denotes the service providers.

1) Compared with OPSR, we use BiLSTM instead of LSTM to perform the training of speech recognition systems (as shown in black in Fig. 1). BiLSTM consists of two unidirectional LSTMs (i.e., the forward LSTM with a forward sequence $\overrightarrow{A}=(x_0,x_1,\ldots,x_t,\ldots)$ and the backward LSTM with a backward sequence $\overleftarrow{A}=(\ldots,x_t,x_{t-1},\ldots,x_0)$ that can be trained in parallel. As we have stated above, utilizing both the left and the right context information of the missed speech data could achieve the accuracy improvement of speech data prediction. Therefore, by combining the context information of speech data, BiLSTM can

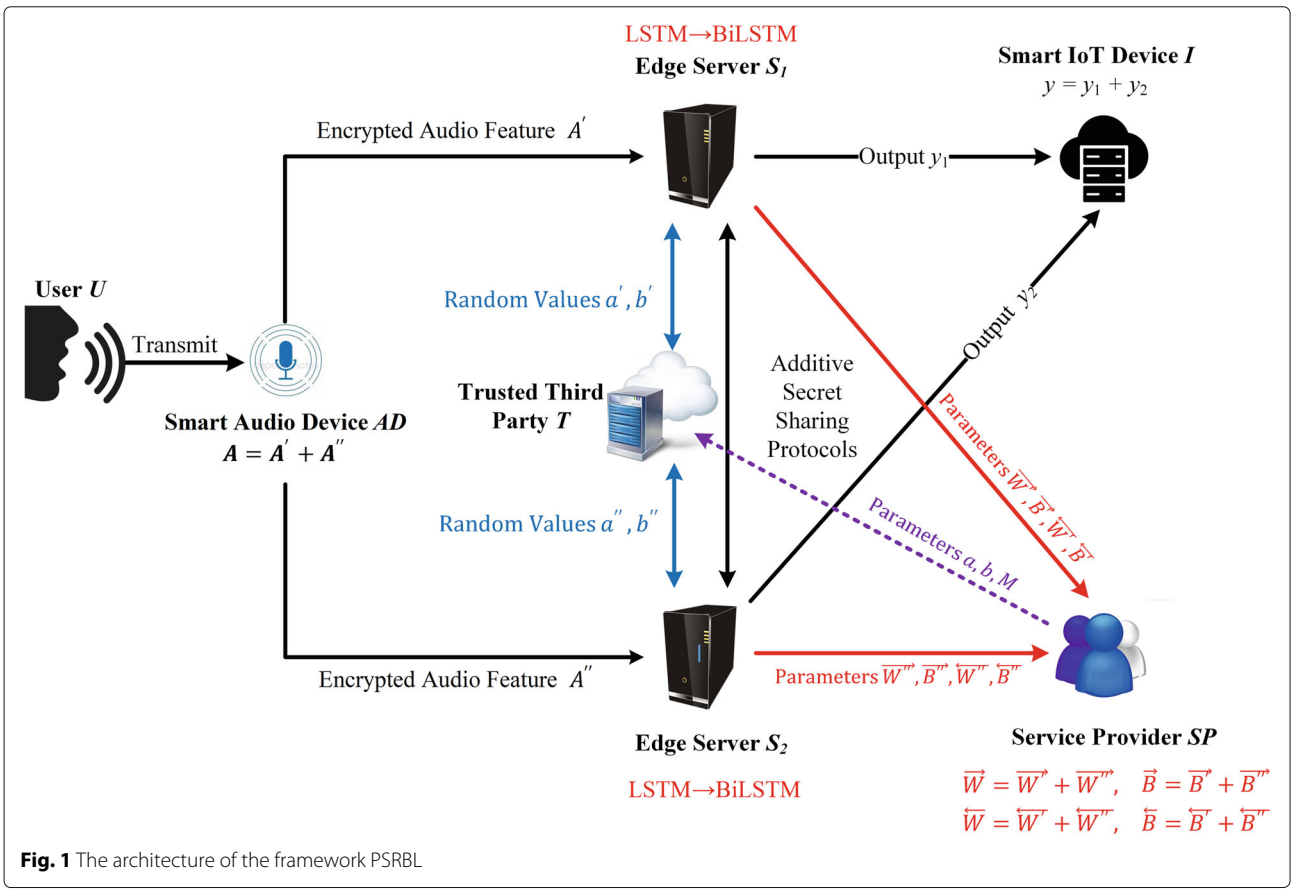


Fig. 1 The architecture of the framework PSRBL

perform effective prediction. The secure forward propagation and the secure back propagation of BiLSTM will be respectively introduced in "Secure forward propagation" and "Secure backward propagation" sections in detail.

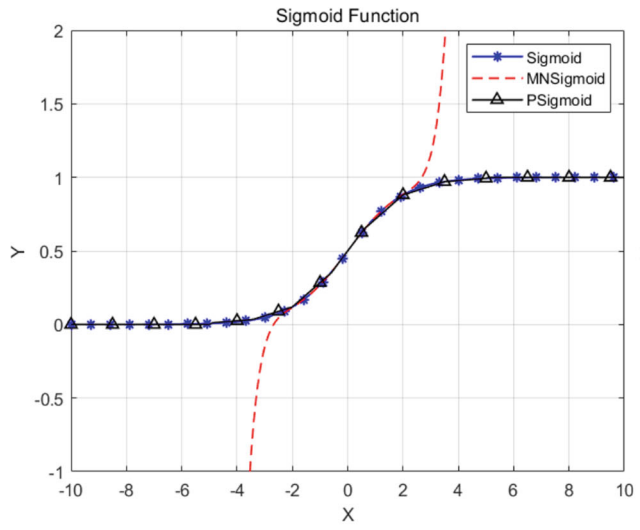
2) New piecewise-linear functions, which are respectively named *PSigmoid* and *PTanh*, are proposed to refit the activation functions with multiple iterations (i.e., *Sigmoid* and *Tanh*) for reducing the time consumption. Compared with *Maclaurin* polynomials, our piecewise-linear functions can greatly reduce the time consumption. The *PSigmoid* and *PTanh* are defined as (1) and (2) respectively. The results of refitting *Sigmoid* by respectively using the piecewise-linear function *PSigmoid* and the activation function *MNSigmoid* used in OPSR are shown in Fig. 2a, and the results of refitting *Tanh* by respectively using the piecewise-linear function *PTanh* and the activation function *MNTanh* used in OPSR are shown in Fig. 2b.

$$PSigmoid(x) = \begin{cases} 0 & x \le -5 \\ 0.015078x + 0.085083 & -5 < x \le -3.5 \\ 0.0599266667x + 0.239053333 & -3.5 < x \le -2 \\ 0.164857143x + 0.448914286 & -2 < x \le -0.6 \\ 0.25x + 0.5 & -0.6 < x \le 0.6 \\ 0.164857143x + 0.551085714 & 0.6 < x \le 2 \\ 0.0599266667x + 0.7609466667 & 2 < x \le 3.5 \\ 0.015078x + 0.91791 & 3.5 < x \le 5 \\ 1 & x > 5 \end{cases} \quad (1)$$

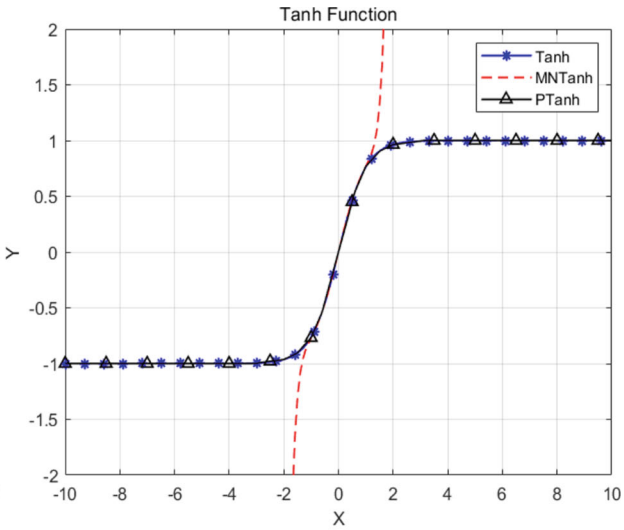
$$PTanh(x) = \begin{cases} 0 & x \le -3 \\ 0.0311x - 0.9018 & -3 < x \le -2 \\ 0.1178x - 0.7284 & -2 < x \le -1.5 \\ 0.2708x - 0.4971 & -1.5 < x \le -1 \\ 0.56975x + 0.19815 & -1 < x \le -0.6 \\ 0.9x & -0.6 < x \le 0.6 \\ 0.56975x + 0.19815 & 0.6 < x \le 1 \\ 0.2708x + 0.4971 & 1 < x \le 1.5 \\ 0.1178x + 0.7284 & 1.5 < x \le 2 \\ 0.0311x + 0.9018 & 2 < x \le 3 \\ 1 & x > 3 \end{cases} \quad (2)$$

After that, *SP* transmits the parameters of piecewise-linear functions and a third-tuple (a, b, M) to T (as shown in purple in Fig. 1), where a and b respectively denote the variable coefficient and the constant term of piecewise-linear functions, and M denotes the piecewise intervals. T first sums the middle values transmitted from both S_1 and S_2 to obtain the interval M, and then randomly divides (a, b) into $\left(a', b'\right)$ and $\left(a'', b''\right)$. Finally, $\left(a', b'\right)$ and $\left(a'', b''\right)$ are respectively sent back to S_1 and S_2 to support their security two-party calculations (as shown in blue in Fig. 1).

From Fig. 2a, we can see that the curve *PSigmoid* excellently refits the curve *Sigmoid*, whereas the curve *MNSigmoid* only performs well in an interval [-2, 2]. Figure 2b shows that the curve *PTanh* excellently refits the curve *Tanh*, whereas the curve *MNTanh* only performs well in an interval [-1, 1].



(a) Fitting Sigmoid function with PSigmoid and MNSigmoid



(b) Fitting Tanh function with PTanh and MNTanh

Fig. 2 The piecewise-linear functions *PSigmoid* and *PTanh* refit original activation functions *Sigmoid* and *Tanh* respectively

Secure forward propagation

For each edge server there is a BiLSTM deployed, and each BiLSTM contains a single **F**orward layer LSTM (F-LSTM in short) and a single **B**ackward layer LSTM (B-LSTM in short). Each single F-LSTM (B-LSTM) handles historical information by deploying a nonlinear function and a large number of linear functions and includes a single forget gate, a single input gate and a single output gate. Given a speech sequence data A=($x_0, x_1, \ldots, x_t, \ldots$), S_1 and S_2 perform the privacy-preserving calculations of the forget gate, the input gate and the output gate by employing a security addition protocol and a secure multiplication protocol. The symbols "$\rightarrow$" and "$\leftarrow$" respectively indicate the calculation processes of F-LSTM and B-LSTM. The superscript symbols "$'$" and "$''$" indicate the calculation processes of neural network on S_1 and S_2 respectively. The symbols as well as the corresponding interpretations used in the paper are shown in Table 1. The secure forward propagation of BiLSTM is shown in Fig. 3.

We combine piecewise-linear functions $PSigmoid(x)$ and $PTanh(x)$ with a security addition protocol and a secure multiplication protocol to construct new secure piecewise-linear functions $SPSigmoid(x)$ and $SPTanh(x)$ respectively, which are treated as activation functions of gates. Since the construction steps of $SPSigmoid(x)$ and $SPTanh(x)$ are similar, we only take $SPSigmoid(x)$ as an example, which is defined as follows.

$$\begin{aligned} SPSigmoid(x) &= a \cdot x + b \\ &= SecAdd\left(SecMul\left(a^{'}, a^{''}, x^{'}, x^{''}\right), b^{'}, b^{''}\right) \end{aligned} \tag{3}$$

where $a^{'}$ and $b^{'}$ respectively denote the coefficient and the constant term of $PSigmoid(x)$ running on S_1; $a^{''}$ and $b^{''}$ respectively denote the coefficient and the constant term of $PSigmoid(x)$ running on S_2.

Forget gate. The forget gate $SPSigmoid(x)$ treats h_{t-1} (the output vector of the previous unit) and x_t (the input vector of the current unit) as input values. For each item in c_{t-1} (the memory vector of the previous unit), $SPSigmoid(x)$ generates a value within [0, 1] to handle the forgetting degree of the previous unit. The relevant calculating processes are shown as follows.

F-LSTM:

$$\begin{aligned} \overrightarrow{u_f} &= \overrightarrow{u_f^{'}} + \overrightarrow{u_f^{''}} \\ &= SecAdd\left(SecMul\left(\overrightarrow{W_f^{'}}, \overrightarrow{W_f^{''}}, \left[\overrightarrow{h_{t-1}^{'}}, x_t^{'}\right], \right.\right. \\ &\quad \left.\left.\left[\overrightarrow{h_{t-1}^{''}}, x_t^{''}\right]\right), \overrightarrow{B_f^{'}} + \overrightarrow{B_f^{''}}\right) \end{aligned} \tag{4}$$

$$\begin{aligned} \overrightarrow{f_t} &= \overrightarrow{f_t^{'}} + \overrightarrow{f_t^{''}} \\ &= SPSigmoid\left(\overrightarrow{u_f}\right) \\ &= SecAdd\left(SecMul\left(\overrightarrow{a^{'}}, \overrightarrow{a^{''}}, \overrightarrow{u_f^{'}}, \overrightarrow{u_f^{''}}\right), \overrightarrow{b^{'}} + \overrightarrow{b^{''}}\right) \end{aligned} \tag{5}$$

B-LSTM:

$$\begin{aligned} \overleftarrow{u_f} &= \overleftarrow{u_f^{'}} + \overleftarrow{u_f^{''}} \\ &= SecAdd\left(SecMul\left(\overleftarrow{W_f^{'}}, \overleftarrow{W_f^{''}}, \left[\overleftarrow{h_{t+1}^{'}}, x_t^{'}\right], \right.\right. \\ &\quad \left.\left.\left[\overleftarrow{h_{t+1}^{''}}, x_t^{''}\right]\right), \overleftarrow{B_f^{'}} + \overleftarrow{B_f^{''}}\right) \end{aligned} \tag{6}$$

$$\begin{aligned} \overleftarrow{f_t} &= \overleftarrow{f_t^{'}} + \overleftarrow{f_t^{''}} \\ &= SPSigmoid\left(\overleftarrow{u_f}\right) \\ &= SecAdd\left(SecMul\left(\overleftarrow{a^{'}}, \overleftarrow{a^{''}}, \overleftarrow{u_f^{'}}, \overleftarrow{u_f^{''}}\right), \overleftarrow{b^{'}} + \overleftarrow{b^{''}}\right) \end{aligned} \tag{7}$$

Input gate. The input gate performs three steps. Step 1), it utilizes $SPSigmoid(x)$ to calculate i_t (which is the filtered input vector of current unit). The relevant calculating processes are defined as follows.

Table 1 Symbols as well as corresponding interpretations used in the paper

Symbol	Interpretation	Symbol	Interpretation
$\rightarrow$	The calculation process of F-LSTM	W_{kh}	Weight matrix of h_{t-1}, where $k \in \{f, i, \tilde{c}, o\}$
$\leftarrow$	The calculation process of B-LSTM	W_{kx}	Weight matrix of x_t, where $k \in \{f, i, \tilde{c}, o\}$
$'$	Parameter running on the edge server S_1	B_k	Bias, where $k \in \{f, i, \tilde{c}, o\}$
$''$	Parameter running on the edge server S_1	u_k	u_k=$W_k \cdot [h_{t-1}, x_t]$+B_k, where $k \in \{f, i, \tilde{c}, o\}$
t	Time node during speech recognition process.	c_t	the neural cell state at time t
k	$k \in \{f, i, \tilde{c}, o\}$	∇W_t	The computational gradient of weight matrix
$\delta_{k,t}$	The calculation error of backward propagation at time t	∇W_{kh}	The computational gradient of weight matrix of h_t
h_t	The output of LSTM at time t	∇W_{kx}	The computational gradient of weight matrix of x_t
W_k	Weight matrix	∇B_k	The computational gradient of bias in back propagation

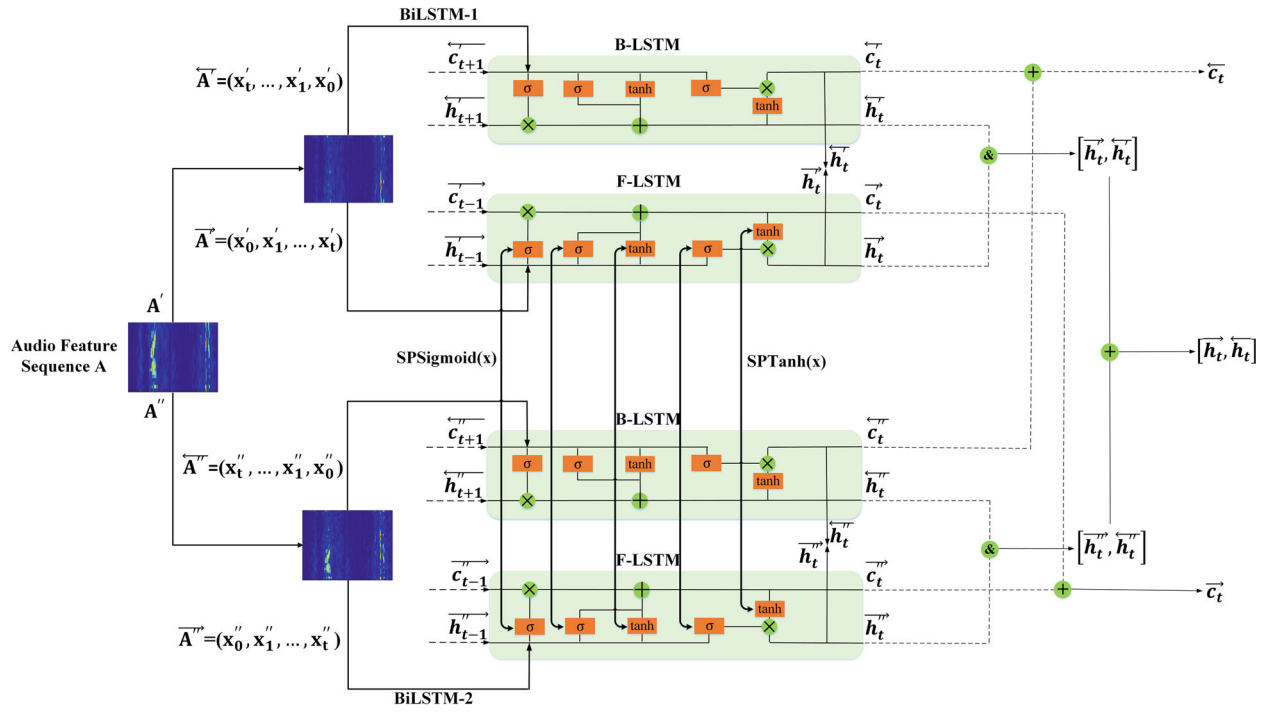


Fig. 3 The secure forward propagation and the secure backward propagation of BiLSTM on S_1 and S_2

F-LSTM:

$$\begin{aligned}\overrightarrow{u_i} &= \overrightarrow{u_i'} + \overrightarrow{u_i''} \\ &= SecAdd\left(SecMul\left(\overrightarrow{W_i'}, \overrightarrow{W_i''}, \left[\overrightarrow{h_{t-1}'}, x_t'\right], \left[\overrightarrow{h_{t-1}''}, x_t''\right]\right), \overrightarrow{B_i'} + \overrightarrow{B_i''}\right)\end{aligned} \tag{8}$$

$$\begin{aligned}\overrightarrow{i_t} &= \overrightarrow{i_t'} + \overrightarrow{i_t''} \\ &= SPSigmoid\left(\overrightarrow{u_i}\right) \\ &= SecAdd\left(SecMul\left(\overrightarrow{a'}, \overrightarrow{a''}, \overrightarrow{u_i'}, \overrightarrow{u_i''}\right), \overrightarrow{b'} + \overrightarrow{b''}\right)\end{aligned} \tag{9}$$

B-LSTM:

$$\begin{aligned}\overleftarrow{u_i} &= \overleftarrow{u_i'} + \overleftarrow{u_i''} \\ &= SecAdd\left(SecMul\left(\overleftarrow{W_i'}, \overleftarrow{W_i''}, \left[\overleftarrow{h_{t+1}'}, x_t'\right], \left[\overleftarrow{h_{t+1}''}, x_t''\right]\right), \overleftarrow{B_i'} + \overleftarrow{B_i''}\right)\end{aligned} \tag{10}$$

$$\begin{aligned}\overleftarrow{i_t} &= \overleftarrow{i_t'} + \overleftarrow{i_t''} \\ &= SPSigmoid\left(\overleftarrow{u_i}\right) \\ &= SecAdd\left(SecMul\left(\overleftarrow{a'}, \overleftarrow{a''}, \overleftarrow{u_i'}, \overleftarrow{u_i''}\right), \overleftarrow{b'} + \overleftarrow{b''}\right)\end{aligned} \tag{11}$$

Step 2), the input gate uses $SPTanh(x)$ to calculate $\tilde{c}_t$ (which is the candidate unit state vector of current unit) for handling how much new information required to be added. The relevant calculating processes are listed as follows.

F-LSTM:

$$\begin{aligned}\overrightarrow{u_{\tilde{c}}} &= \overrightarrow{u_{\tilde{c}}'} + \overrightarrow{u_{\tilde{c}}''} \\ &= SecAdd\left(SecMul\left(\overrightarrow{W_{\tilde{c}}'}, \overrightarrow{W_{\tilde{c}}''}, \left[\overrightarrow{h_{t-1}'}, x_t'\right], \left[\overrightarrow{h_{t-1}''}, x_t''\right]\right), \overrightarrow{B_{\tilde{c}}'} + \overrightarrow{B_{\tilde{c}}''}\right)\end{aligned} \tag{12}$$

$$\begin{aligned}\overrightarrow{\tilde{c}_t} &= \overrightarrow{\tilde{c}_t'} + \overrightarrow{\tilde{c}_t''} \\ &= SPmoid\left(\overrightarrow{u_{\tilde{c}}}\right) \\ &= SecAdd\left(SecMul\left(\overrightarrow{a'}, \overrightarrow{a''}, \overrightarrow{u_{\tilde{c}}'}, \overrightarrow{u_{\tilde{c}}''}\right), \overrightarrow{b'} + \overrightarrow{b''}\right)\end{aligned} \tag{13}$$

B-LSTM:

$$\begin{aligned}\overleftarrow{u_{\tilde{c}}} &= \overleftarrow{u_{\tilde{c}}^{'}} + \overleftarrow{u_{\tilde{c}}^{''}} \\ &= SecAdd\left(SecMul\left(\overleftarrow{W_{\tilde{c}}^{'}}, \overleftarrow{W_{\tilde{c}}^{''}}, \left[\overleftarrow{h_{t+1}^{'}}, x_t^{'}\right],\right.\right. \\ &\left.\left.\left[\overleftarrow{h_{t+1}^{''}}, x_t^{''}\right]\right), \overleftarrow{B_{\tilde{c}}^{'}} + \overleftarrow{B_{\tilde{c}}^{''}}\right)\end{aligned} \tag{14}$$

$$\begin{aligned}\overleftarrow{\tilde{c}_t} &= \overleftarrow{\tilde{c}_t^{'}} + \overleftarrow{\tilde{c}_t^{''}} \\ &= SPmoid\left(\overleftarrow{u_{\tilde{c}}}\right) \\ &= SecAdd\left(SecMul\left(\overleftarrow{a^{'}}, \overleftarrow{a^{''}}, u_{\tilde{c}}^{'}, u_{\tilde{c}}^{''}\right), \overleftarrow{b^{'}} + \overleftarrow{b^{''}}\right)\end{aligned} \tag{15}$$

and Step 3), by combining with a secure addition function and a secure multiplication function, the input gate employs $\tilde{c}_{t-1}$, i_t and $\tilde{c}_t$ to update the unit state, which are calculated as follows.

F-LSTM:

$$\begin{aligned}\overrightarrow{c_t} &= \overrightarrow{c_t^{'}} + \overrightarrow{c_t^{''}} \\ &= SecAdd\left(SecMul\left(\overrightarrow{f_t^{'}}, \overrightarrow{f_t^{''}}, \overrightarrow{c_{t-1}^{'}}, \overrightarrow{c_{t-1}^{''}}\right),\right. \\ &\left.SecMul\left(\overrightarrow{i_t^{'}}, \overrightarrow{i_t^{''}}, \overrightarrow{\tilde{c}_t^{'}}, \overrightarrow{\tilde{c}_t^{''}}\right)\right)\end{aligned} \tag{16}$$

B-LSTM:

$$\begin{aligned}\overleftarrow{c_t} &= \overleftarrow{c_t^{'}} + \overleftarrow{c_t^{''}} \\ &= SecAdd\left(SecMul\left(\overleftarrow{f_t^{'}}, \overleftarrow{f_t^{''}}, \overleftarrow{c_{t+1}^{'}}, \overleftarrow{c_{t+1}^{''}}\right),\right. \\ &\left.SecMul\left(\overleftarrow{i_t^{'}}, \overleftarrow{i_t^{''}}, \overleftarrow{\tilde{c}_t^{'}}, \overleftarrow{\tilde{c}_t^{''}}\right)\right)\end{aligned} \tag{17}$$

Output gate. The input values of the output gate are the calculated results coming from the input gate and the forget gate, which means the output of BiLSTM is influenced by both long-term memory and the current input value. The output gate first utilizes W_o (the output weight matrix) and B_o (the output bias term) to calculate o_t (the current output vector of the output gate) by using following equations.

F-LSTM:

$$\begin{aligned}\overrightarrow{u_o} &= \overrightarrow{u_o^{'}} + \overrightarrow{u_o^{''}} \\ &= SecAdd\left(SecMul\left(\overrightarrow{W_o^{'}}, \overrightarrow{W_o^{''}}, \left[\overrightarrow{h_{t-1}^{'}}, x_t^{'}\right],\right.\right. \\ &\left.\left.\left[\overrightarrow{h_{t-1}^{''}}, x_t^{''}\right]\right), \overrightarrow{B_o^{'}} + \overrightarrow{B_o^{''}}\right)\end{aligned} \tag{18}$$

$$\begin{aligned}\overrightarrow{o_t} &= \overrightarrow{o_t^{'}} + \overrightarrow{o_t^{''}} \\ &= SPSigmoid\left(\overrightarrow{u_o}\right) = \\ &SecAdd\left(SecMul\left(\overrightarrow{a^{'}}, \overrightarrow{a^{''}}, \overrightarrow{u_o^{'}}, \overrightarrow{u_o^{''}}\right), \overrightarrow{b^{'}} + \overrightarrow{b^{''}}\right)\end{aligned} \tag{19}$$

B-LSTM:

$$\begin{aligned}\overleftarrow{u_o} &= \overleftarrow{u_o^{'}} + \overleftarrow{u_o^{''}} \\ &= SecAdd\left(SecMul\left(\overleftarrow{W_o^{'}}, \overleftarrow{W_o^{''}}, \left[\overleftarrow{h_{t+1}^{'}}, x_t^{'}\right],\right.\right. \\ &\left.\left.\left[\overleftarrow{h_{t+1}^{''}}, x_t^{''}\right]\right), \overleftarrow{B_o^{'}} + \overleftarrow{B_o^{''}}\right)\end{aligned} \tag{20}$$

$$\begin{aligned}\overleftarrow{o_t} &= \overleftarrow{o_t^{'}} + \overleftarrow{o_t^{''}} \\ &= SPSigmoid\left(\overleftarrow{u_o}\right) \\ &= SecAdd\left(SecMul\left(\overleftarrow{a^{'}}, \overleftarrow{a^{''}}, \overleftarrow{u_o^{'}}, \overleftarrow{u_o^{''}}\right), \overleftarrow{b^{'}} + \overleftarrow{b^{''}}\right)\end{aligned} \tag{21}$$

Then by combining with $SPSigmoid(x)$ and $SPTanh(x)$, h_t (the output vector of current unit) of BiLSTM is calculated according to o_t and c_t. The relevant calculating processes are defined by following equations.

F-LSTM:

$$\begin{aligned}\overrightarrow{h_t} &= \overrightarrow{h_t^{'}} + \overrightarrow{h_t^{''}} = \overrightarrow{o_t} \cdot SPTanh\left(\overrightarrow{c_t}\right) \\ &= SecMul\left(\overrightarrow{o_t^{'}}, \overrightarrow{o_t^{''}}, SecAdd\left(SecMul\left(\overrightarrow{a^{'}}, \overrightarrow{a^{''}}, \overrightarrow{c_t^{'}}, \overrightarrow{c_t^{''}}\right), \overrightarrow{b^{'}}, \overrightarrow{b^{''}}\right)\right)\end{aligned} \tag{22}$$

B-LSTM:

$$\begin{aligned}\overleftarrow{h_t} &= \overleftarrow{h_t^{'}} + \overleftarrow{h_t^{''}} = \overleftarrow{o_t} \cdot SPTanh\left(\overleftarrow{c_t}\right) \\ &= SecMul\left(\overleftarrow{o_t^{'}}, \overleftarrow{o_t^{''}}, SecAdd\left(SecMul\left(\overleftarrow{a^{'}}, \overleftarrow{a^{''}}, \overleftarrow{c_t^{'}}, \overleftarrow{c_t^{''}}\right), \overleftarrow{b^{'}}, \overleftarrow{b^{''}}\right)\right)\end{aligned} \tag{23}$$

In the entire process of secure forward propagation of BiLSTM, $\left[\overrightarrow{h_t}, \overleftarrow{h_t}\right]$ is the input vector of the output layer at time t, where $\overrightarrow{h_t} = \overrightarrow{h_t^{'}} + \overrightarrow{h_t^{''}}$ and $\overleftarrow{h_t} = \overleftarrow{h_t^{'}} + \overleftarrow{h_t^{''}}$. Note that $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$ are calculated by using F-LSTM and B-LSTM respectively.

Secure backward propagation

The basis of the secure backward propagation of BiLSTM is **B**ack **P**ropagation **T**rough **T**ime (BPTT) algorithm, and there is no complex derivative calculation in the secure backward propagation of BiLSTM in this paper, since the original activation functions $Sigmoid(x)$ and $Tanh(x)$ are refitted by $SPSigmoid(x)$ and $SPTanh(x)$ respectively, where $SPSigmoid^{'}(x)=a$ and $SPTanh^{'}(x)=a$. That is, the training process of the secure backward propagation of

Table 2 Comparison of error and time according to different sigmoid piecewise-linear functions

The number of piecewises	3	5	7	8	9	10
The error of forward fropagation	3.43×10^{-2}	5.74×10^{-8}	7.95×10^{-12}	2.36×10^{-14}	2.11×10^{-14}	1.94×10^{-14}
Time(ms)	52.936	65.167	82.728	97.539	113.185	125.479

BiLSTM can be performed by directly utilizing the secure addition protocol and the secure multiplication protocol, which is similar to that of its secure forward propagation.

Based on the trusted third party T, the secure BTPP is performed by utilizing the results from security forward propagation to obtain the error of the loss function. After that, sending the error back to edge servers to update the corresponding weight matrix and bias items. BiLSTM iteratively runs the process until the error converges. Since the calculation process of back propagation in B-LSTM is the same as those of F-LSTM, we only take F-LSTM as an example to introduce details below.

F-LSTM:

$$\begin{aligned}\overrightarrow{\delta_{t-1}} &= \overrightarrow{\delta_{(o,t)}}\overrightarrow{W_{oh}} + \overrightarrow{\delta_{(f,t)}}\overrightarrow{W_{fh}} \\ &+ \overrightarrow{\delta_{(i,t)}}\overrightarrow{W_{ih}} + \overrightarrow{\delta_{(\tilde{c},t)}}\overrightarrow{W_{ih}} = f\left(\overrightarrow{h_{t-1}}\right)\end{aligned} \tag{24}$$

where $\overrightarrow{\delta_{t-1}}$ denotes the total error vector at time *t-1*, $\overrightarrow{\delta_{(o,t)}}, \overrightarrow{\delta_{(f,t)}}, \overrightarrow{\delta_{(i,t)}} and \overrightarrow{\delta_{(\tilde{c},t)}}$ are partial error vectors, which are respectively calculated as follows.

$$\overrightarrow{\delta'_{(o,t)}}, \overrightarrow{\delta''_{(o,t)}} = \overrightarrow{\delta_t} \cdot SPTanh\left(\overrightarrow{c_t}\right) \cdot \overrightarrow{o_t} \cdot \left(1 - \overrightarrow{o_t}\right) \tag{25}$$

$$\overrightarrow{\delta'_{(f,t)}}, \overrightarrow{\delta''_{(f,t)}} = \overrightarrow{\delta_t} \cdot \overrightarrow{o_t} \cdot SPTanh'\left(\overrightarrow{c_t}\right) \cdot \overrightarrow{f_t} \cdot \left(1 - \overrightarrow{f_t}\right) \tag{26}$$

$$\overrightarrow{\delta'_{(i,t)}}, \overrightarrow{\delta''_{(i,t)}} = \overrightarrow{\delta_t} \cdot \overrightarrow{o_t} \cdot SPTanh'\left(\overrightarrow{c_t}\right) \cdot \overrightarrow{\tilde{c}_t} \cdot \overrightarrow{i_t} \cdot \left(1 - \overrightarrow{i_t}\right) \tag{27}$$

$$\overrightarrow{\delta'_{(\tilde{c},t)}}, \overrightarrow{\delta''_{(\tilde{c},t)}} = \overrightarrow{\delta_t} \cdot \overrightarrow{o_t} \cdot SPTanh'\left(\overrightarrow{c_t}\right) \cdot \overrightarrow{i_t} \cdot \left(1 - \overrightarrow{\tilde{c}_t^2}\right) \tag{28}$$

$$SPTanh'\left(\overrightarrow{c_t}\right) = 1 - SPTanh\left(\overrightarrow{c_t}\right)^2 \tag{29}$$

Initializing the total error vector at time t to 1, i.e., $\overrightarrow{\delta_t}$ =1, and the total error vector before time t can be obtained by (24). Given the partial error vectors at each time, $\overrightarrow{h_{t-1}}$, $\overrightarrow{W_k}$ and $\overrightarrow{B_k}$. Meanwhile, by using the symbol ∇ to denote the gradient, we can obtain an equation:

$$\begin{aligned}\overrightarrow{\nabla W_k} &= \left[\overrightarrow{\nabla W_{kh}}, \overrightarrow{\nabla W_{kx}}\right] \\ &= \left[\overrightarrow{\nabla W'_{kh}} + \overrightarrow{\nabla W''_{kh}}, \overrightarrow{\nabla W'_{kx}} + \overrightarrow{\nabla W''_{kx}}\right]\end{aligned} \tag{30}$$

In addition, the corresponding derivations are shown as follows.

$$\overrightarrow{\nabla W'_{kh}}, \overrightarrow{\nabla W''_{kh}} = \sum_{j=1}^{t} SecMul\left(\overrightarrow{\delta_{(k,j)}}, \overrightarrow{h_{t-1}}\right) \tag{31}$$

$$\overrightarrow{\nabla W'_{kx}}, \overrightarrow{\nabla W''_{kx}} = SecMul\left(\overrightarrow{\delta_{(k,j)}}, \overrightarrow{x_t}\right) \tag{32}$$

$$\overrightarrow{\nabla B'_k}, \overrightarrow{\nabla B''_k} = \sum_{j=1}^{t} \overrightarrow{\delta_{(k,j)}} \tag{33}$$

Using α to denote the learning rate of BiLSTM, and supposing that α is public and available, the weight matrix and the bias term can be updated by using the gradient values, which are calculated as follows.

$$\overrightarrow{W'_{(new,k)}}, \overrightarrow{W''_{(new,k)}} = \overrightarrow{W_{(old,k)}} - \overrightarrow{\nabla W_k} \cdot \alpha \tag{34}$$

$$\overrightarrow{B'_{(new,k)}}, \overrightarrow{B''_{(new,k)}} = \overrightarrow{B_{(old,k)}} - \overrightarrow{\nabla B_k} \cdot \alpha \tag{35}$$

Analyses for correctness and security analysis

Correctness. As shown in Fig. 2, the piecewise-linear functions $PSigmoid(x)$ and $PTanh(x)$ perform well. Theoretically, $PSigmoid(x)$ and $PTanh(x)$ can infinitely approach the original activation functions $Sigmoid(x)$ and $Tanh(x)$. In addition, the foundations of secure piecewise-linear functions $SPSigmoid(x)$ and $SPTanh(x)$ are the secure addition protocol and the secure multiplication protocol of the addition secret sharing protocol whose correctness has been proved in [14].

Security Analysis. In the processes of forward propagation and back propagation, on the one hand, all input vectors of secure piecewise-linear functions $SPSigmoid(x)$ and $SPTanh(x)$ are the calculated results of the secure

Table 3 Comparison of error and time according to different tanh piecewise-linear functions

The number of piecewises	3	5	7	9	11	12
The error of forward fropagation	1.14×10^{-2}	5.47×10^{-8}	8.51×10^{-12}	2.37×10^{-14}	2.26×10^{-14}	2.16×10^{-14}
Time(ms)	43.573	54.947	67.695	82.421	97.647	112.193

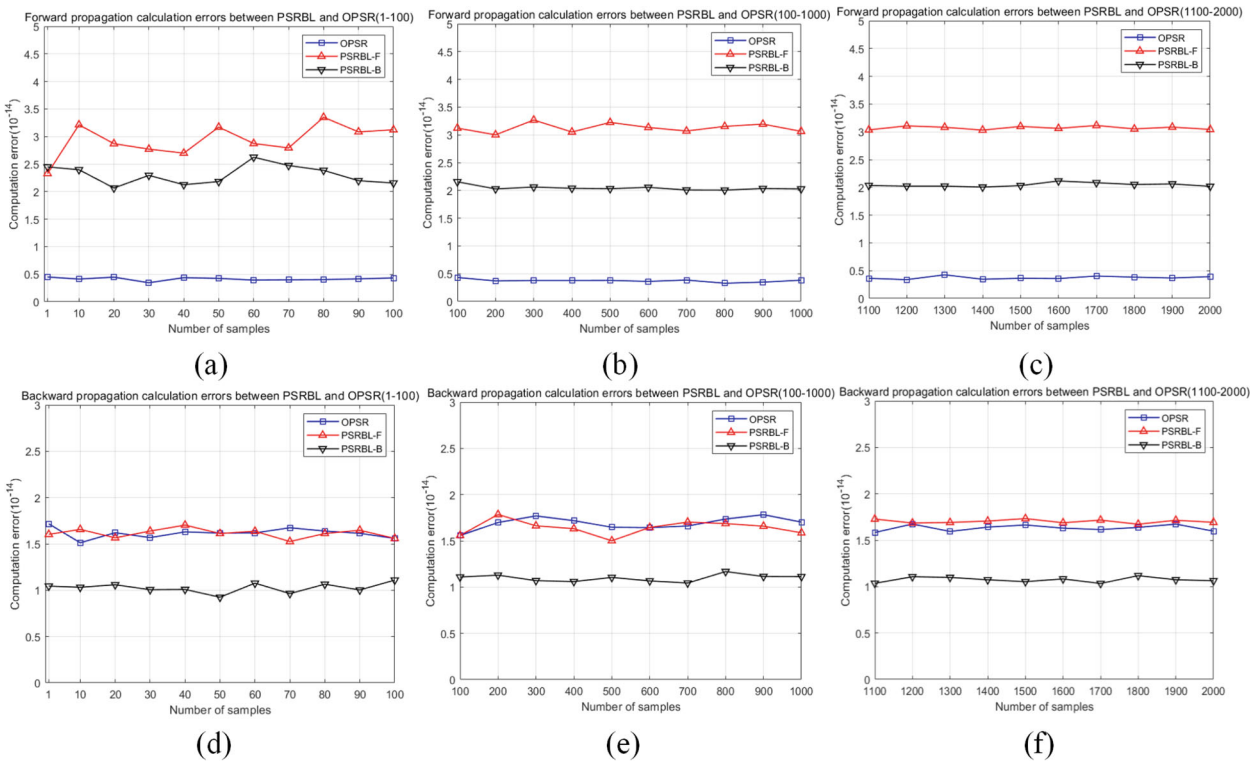


Fig. 4 The computation errors of forward propagation and backward propagation

addition protocol and the secure multiplication protocol. On the other hand, all output vectors of the forget gate, the input gate and the output gate are calculated by $SPSigmoid(x)$ and $SPTanh(x)$. Since the security of the secure addition protocol and the secure multiplication protocol has been demonstrated in [14], the processes of forward propagation and back propagation are secure.

Experiments

In this section, we conduct multiple experiments to evaluate the performance of our PSRBL. We focus in answering the following two research questions.

(1) *Question 1.* Can the proposed piecewise-linear activation functions achieve the accuracy improvement of our PSRBL?
(2) *Question 2.* Compared with existing state-of-the-art peers, does our PSRBL perform better?

Experimental dataset and environment

We utilize the state-of-the-art framework OPSR as comparison. OPSR is implemented in *Python*3 and *Numpy* in [14]. The adopted speech dataset is a part of TIMIT corpus, which has 123 features coming from the Fourier transform-based filter library [17]. To make fair

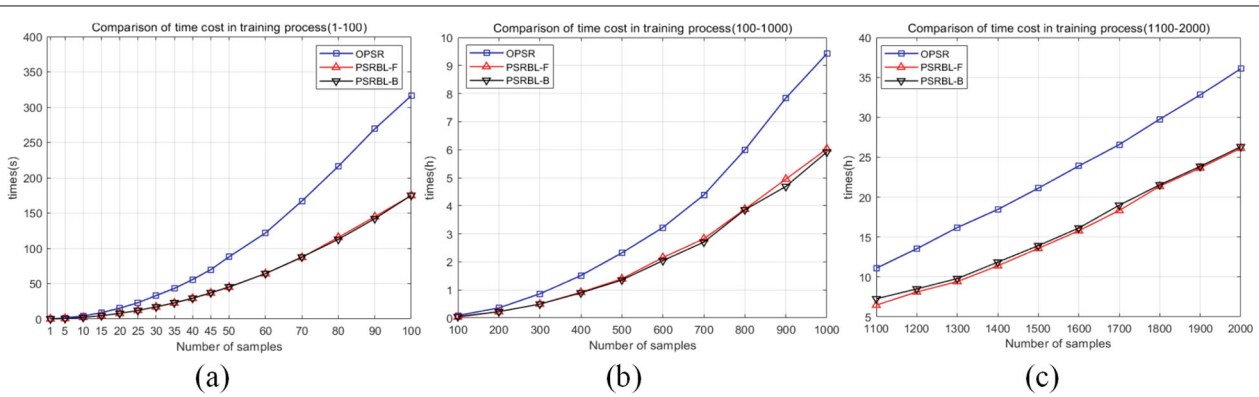


Fig. 5 The computation time of frameworks PSRBL and OPSR

Table 4 Runtime of privacy-preserving LSTM for each training sample

State	Our Scheme(ms)		OPSR Scheme(ms)	
	Initialization	Computation	Initialization	Computation
Forward Propagation	2.88	24.85	2.11	107.11
Backward Propagation	1.02	26.95	1.10	60.06

comparisons, the performance of our PSRBL is also evaluated on the same dataset. Given a data length l=64, 2000 processed speech sequence data with 123 features applied on a neural network with 80 neurons, and the time step is set to 8. Therefore, the size of the weight matrix W_k is 203 $\times$80, where $k \in \{f, i, \tilde{c}, o\}$, and the output matrix and the offset terms are 8 $\times$80 and 1 $\times$80 respectively. In addition, PSRBL is also implemented by *Python*3 and *Numpy*, and follows the same experimental parameter settings that OPSR used. Note that OPSR uses the *McLaughlin* polynomial and the *Newton* iteration method, where the number of iterations is set to 10. However, our PSRBL does not need iterations, since its activation functions are refitted by the piecewise-linear functions. In our experiments, edge servers have the same configuration, which are Inter(R)Core(TM) *i*5-7500 CPU @3.40GHz and 8.00GB of memory.

Performance of the piecewise-linear activation functions (Question 1)

Since the piecewise-linear activation functions achieve an accuracy improvement of framework PSRBL, we first conduct multiple experiments to evaluate the performance of piecewise-linear activation functions with different numbers.

The experimental results of *Sigmoid* piecewise-linear fitting activation function *PNSigmoid* are shown in Table 2. With the increment of the piecewise number, the error values of the forward propagation of both original and secure LSTMs continuously reduce. When the piecewise number is 8, the error value reached 10^{-14}, which can be ignored. That is, as the piecewise number increases continuously, the change of the error value can be ignored but the time consumption significantly increases. Therefore, the piecewise number of fitting *Sigmoid* activation function is set to 8.

The results of *Tanh* piecewise-linear fitting activation function *PNTanh* are shown in Table 3. When the piecewise number is 11, the error value reaches 10^{-14}, which can be ignored. After that, as the piecewise number continuously increases, the time consumption significantly increases, so as to the piecewise number of fitting *Tanh* activation function is set to 11.

Performance comparisons (Question 2)

The experimental comparisons are shown in Figs. 4 and 5. Note that PSRBL-F and PSRBL-B in the figures respectively correspond to F-LSTM and B-LSTM of BiLSTM (the key component of our PSRBL).

From Fig. 4, we can see that with the increment of the number of samples, 1) the calculation errors of the forward propagation of PSRBL-F, PSRBL-B and OPSR are all as low as 10^{-14} (as shown in Fig. 4a); and 2) the calculation errors of the backward propagation of PSRBL-F, PSRBL-B and OPSR are all as low as 10^{-14} too (as shown in Fig. 4b). Therefore, we can conclude that the frameworks PSRBL and OPSR have the same performance in the training process of the neural network with privacy-preserving.

As shown in Fig. 5, with the increment of the number of samples, the time consumptions of PSRBL-F, PSRBL-B and OPSR increase too. However, both PSRBL-F and PSRBL-B are much more efficient than OPSR. In the case where the number of samples is less than 100, the time consumptions of both PSRBL-F and PSRBL-B is half of that of OPSR (as shown in Fig. 5a). In the case where the number of samples is between 100 and 1000, OPSR takes one and a half times as long as both PSRBL-F and PSRBL-B do (as shown in Fig. 5b). In the case where the number of samples is between 1000 and 2000, OPSR still takes more time consumption as long as both PSRBL-F and PSRBL-B do (as shown in Fig. 5c).

To sum up, we can conclude that PSRBL significantly reduces the time of both the training and the recognition of the speech model. The reason is that the secure activation functions in PSRBL do not require iterations.

Table 5 Forward propagation runtime of each LSTM gate for each training sample

Gate	Our Scheme(ms)		OPSR Scheme(ms)	
	Initialization	Computation	Initialization	Computation
Forget Gate	0.76	7.52	0.63	23.89
Input Gate	1.39	10.92	1.05	44.02
Output Gate	0.72	6.41	0.43	39.20

Table 6 Message size of multiplication and addition using homomorphic encryption

Homomorphic encryption	Additive homomorphic encryption	Multiplicative homomorphic encryption
Plaintext(Byte)	8	8
Ciphertext(Byte)	154	307

The time spent on a single iteration of a single secure activation function in OPSR, denoted as t_s, is the sum of the time of running one-time secure addition protocol and one-time secure multiplication protocol. In the experiments, the number of iterations that secure activation functions required in OPSR is 10, which means the total time spent on a single secure activation function is $10t_s$. PSRBL uses piecewise-linear functions instead of secure activation functions to avoid iterations. Supposes that T needs t_p to construct the piecewise intervals, the total time spent on a single secure activation function is t_s+t_p, which means the time consumption of PSRBL depends on t_p.

Table 4 shows the time consumptions (including initialization time and calculation time of a single sample) of forward and backward calculations. The initialization time refers to 1) the initialization of weight matrix and calculation vectors for each bias term, and 2) the initialization of three gates of LSTM. It can be seen that the initialization time of PSRBL is roughly the same as that of OPSR. However, for the time consumption of forward calculation, OPSR takes 4.5 times as long as PSRBL does, and for the time consumption of backward calculation, OPSR takes 2.3 times as long as PSRBL does.

Table 5 shows the time consumptions of three gates of forward and backward calculations. It can be seen that the initialization time of PSRBL is roughly the same as that of OPSR. For the time consumptions of the forget gate, the input gate and the output gate, OPSR respectively takes 3, 4 and 6 times as long as PSRBL does.

As shown in Table 6, the privacy-preserving homomorphic encryption-based scheme needs to encrypt the data at first, which extends the data size, so as to greatly increase the overhead of communication storage during the training process of BiLSTM and applications of intelligent voice.

Table 7 shows that the framework based on additive secret sharing performs better than the homomorphic encryption-based framework. In order to process one frame of audio, the homomorphic encryption-based privacy protection Gaussian mixture model (GMM) scheme [11] requires 616.759 ms, the time consumption is almost 6 times that of PSRBL, and the overhead of communication storage is more than 40 times that of PSRBL. In addition, compared with Hidden Markov Model (HMM) [10] and OPSR based on the addition secret sharing, BiLSTM-based models achieve higher accuracy. Compared with OPSR based on the addition secret sharing, PSRBL can significantly reduce the time consumption, which means it can not only save the training time for the entire model, but also improve the response time during speech recognition.

Related work

Edge computing is a popular computing paradigm with the aim to minimize the delay between end-users and the cloud, and many applications in our daily life have been promoted and developed, such as QoS prediction [18, 19]. However, it is well known that edge computing suffers from capacity limitations. Thus, the edge-cloud computing paradigm is proposed to balance computing resources and low latency. Speech recognition [3] is one of the most important applications.

In the early days, speech recognition research mainly used Hidden Markov Model (HMM) [20, 21]. Lee et al. [22] proposed a co-occurrence smoothing algorithm that enables accurate speech recognition on a minimal training dataset. Nevertheless, HMM neglects the long-term dependence relations between speech data. Lipton et al. [23] showed that Recurrent Neural Networks (RNN) could effectively solve the above problem. Differ from the sequential structure of Convolutional Neural Network (CNN) [24], RNN forms a complex recurrent chain structure through input layer, hidden layer and output layer. Gers et al. [25] improved the LSTM network structure by adding a forget gate and peepholes, since RNN becomes unreliable under complex application environments because of the long-term dependencies

Table 7 Comparison of runtime and message size with other schemes

	Runtime Per Sample(ms)	Message Size per Sample(MB)
Our Scheme	97.647	4.617
OPSR Scheme	227.576	4.873
GMM with HE [11]	616.759	95.2
HMM with 2PC [10]	502.351	13.22

problem [26]. However, LSTM neglects the right context information of speech data, which results in loss of semantic information. Hence, Alex Graves et al. [27] proposed a bidirectional long-term and short-term memory neural network. Unlike the way of training from left to right that used in LSTM, Alex Graves et al. made the use of the left and the right context information to train the speech recognition model. Bin et al. [28] proposed a video captioning framework based on bidirectional long-term memory and soft attention mechanisms to enhance the ability of recognizing persistent motion in the video.

In the development of speech recognition technology, most speech recognition systems are deployed on cloud and (or) edge servers, and meanwhile the speech data is stored and transmitted in cleartext. That is, speech data with private information can be maliciously collected and analyzed [9]. Therefore, it is a feasible solution to protect speech data by using encryption algorithms. CryptoNets [29] is a neural network that can be trained by ciphertext. Some studies use Homomorphic Encryption (HE) to protect data privacy, such as MiniONN [30]. For example, Zhang et al. used BGV to achieve the encryption of the private data [31]; Yilmaz et al. utilized a partially homomorphic cryptosystem as the element of the proposed privacy-preserving solutions [32]. HE-based methods are time-consuming and require high capacity memory. Differential privacy is another data privacy-preserving method. However, it decreases the accuracy of speech recognition because of low data availability [11, 17]. SMC can effectively improve the availability of encrypted data by combining edge servers [4]. Huang et al. [33] proposed a SMC-based lightweight framework to protect data privacy. Ma et al. [14] proposed a privacy-preserving speech recognition framework based on LSTM and SMC to achieve privacy-preserving.

Conclusion

This paper proposes a novel **P**rivacy-preserving **S**peech **R**ecognition framework using the **B**idirectional **L**ong short-term memory neural network (PSRBL). PSRBL makes full use of the left and the right context of speech data to improve the accuracy of speech recognition, and employs piecewise-linear functions to refit the original activation functions for reducing training and recognition time. In addition, PSRBL achieves the privacy-preserving of speech data during speech recognition based on SMC. The experimental results show that PSRBL outperforms the existing approaches.

Abbreviations

RNN: Recurrent Neural Network; CNN: Convolutional Neural Network; BiLSTM: Bidirectional Long Short-Term Memory; IoTs: Internet of things; SMC: Secure Multi-Party Computing; LSTM: Long Short-Term Memory; F-LSTM: forward layer LSTM; B-LSTM: backward layer LSTM; BPTT: Back Propagation Trough Time

Acknowledgements

The author is grateful to the Key Laboratory of Intelligent Computing and Signal Processing of Ministry of Education for funding this research.

Authors' contributions

All authors take part in the discussion of the work described in this paper. All authors have worked equally when writing this paper. All author(s) read and approved the final manuscript.

Authors' information

Qingren Wang is a lecturer at the School of Computer Science and Technology at the Anhui University of China. Chuankai Feng is a master candidate at the School of Computer Science and Technology at the Anhui University of China. Yan Xu is an associate professor at the School of Computer Science and Technology at the Anhui University of China. Hong Zhong is a professor at the School of Computer Science and Technology at the Anhui University of China. Victor S Sheng is an associate professor of the Department of Computer Science with the Texas Tech University.

Funding

This work is partially supported by the National Natural Science Foundation of China under Grant (U1936220,61702005), the National Key R&D Program of China under Grant (2019YFB1704101), the Natural Science Foundation of Anhui Province (1808085MF197, 2008085QF307), the Anhui Foundation for Science and Technology Major Project under Grant (18030901034), and the 2019 Anhui University Collaborative Innovation Project (GXXT-2019-013).

Availability of data and materials

The data has been gathered from published research papers and articles that are mentioned in "Experiments" section.

Competing interests

The authors declare no conflict of interest.

Author details

[1]Key Laboratory of Intelligent Computing and Signal Processing of Ministry of Education, School of Computer Science and Technology, Anhui University, 111 Jiulong Road, Hefei, China. [2]Department of Computer Science, Texas Tech University, 1508 Knoxville Avenue, Lubbock, USA.

Received: 15 January 2020 Accepted: 25 June 2020
Published online: 08 July 2020

References

1. Alaa M, Zaidan A, Zaidan B, Talal M, Mat Kiah M (2017) A review of smart home applications based on internet of things. J Netw Comput Appl 97:48–65
2. Paden B, Cap M, Yong S, Yershov D, Frazzoli E (2016) A survey of motion planning and control techniques for self-driving urban vehicles. IEEE Trans Intell Veh 1(1):33–55
3. Graves A, Mohamed A, Hinton G (2013) Speech recognition with deep recurrent neural networks. In: Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing: 26-31 May 2013. IEEE, Vancouver
4. Shi W, Cao J, Zhang Q, Li Y, Xu L (2016) Edge computing: Vision and challenges. IEEE Internet Things J 3(5):637–646
5. Zhang Y, Cui G, Deng S, Chen F, Wang Y, He Q Efficient query of quality correlation for service composition. IEEE Trans Serv Comput. https://doi.org/10.1109/TSC.2018.2830773 (in press)
6. Zhou J, Sun J, Cong P, Liu Z, Wei T, Zhou X, Hu S (2019) Security-critical energy-aware task scheduling for heterogeneous real-time mpsocs in iot. IEEE Trans Serv Comput. https://doi.org/10.1109/TSC.2019.2963301 (in press)
7. Gong W, Qi L, Xu Y (2018) Privacy-aware multidimensional mobile service quality prediction and recommendation in distributed fog environment. Wirel Commun Mob Comput 2018:1–8

8. Xu Y, Qi L, Dou W, Yu J (2017) Privacy-preserving and scalable service recommendation based on simhash in a distributed cloud environment. Complexity 2017:1–9
9. Hossain M (2016) Patient state recognition system for healthcare using speech and facial expressions. J Med Syst 40(12):272
10. Smaragdis P, Shashanka M (2007) A framework for secure speech recognition. IEEE Trans Audio Speech Lang Process 15(4):1404–1413
11. Pathak M, Raj B (2012) Privacy-preserving speaker verification and identification using gaussian mixture models. IEEE Trans Audio Speech Lang Process 21(2):397–406
12. Abadi M, Chu A, Goodfellow I, McMahan H, Mironov I, Talwar K, Zhang L (2016) Deep learning with differential privacy. In: Proceedings of ACM SIGSAC Conference on Computer and Communications Security: 24-28 October 2016. ACM, Vienna
13. Cramer R, Damgard I, Nielsen J (2015) Secure Multiparty Computation and Secret Sharing. Cambridge University Press, New York, NY, United States. https://doi.org/10.1017/CBO9781107337756
14. Ma Z, Liu Y, Liu X, Ma J, Li F (2019) Privacy-preserving outsourced speech recognition for smart iot devices. IEEE Internet Things J 6(5):8406–8420
15. Graves A, Jaitly N, Mohamed A (2013) Hybrid speech recognition with deep bidirectional lstm. In: Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding 8- 12 Dec. 2013. IEEE, Olomouc. pp 8–12
16. Bin Y, Yang Y, Shen F, Xie N, Shen H, Li X (2018) Describing video with attention-based bidirectional lstm. IEEE Trans Dependable Secure Comput 49(7):2631–2641
17. Zhu T, Li G, Zhou W, Y P (2017) Differentially private data publishing and analysis: A survey. IEEE Trans Knowl Data Eng 29(8):1619–1638
18. Zhang Y, Yin C, Wu Q, He Q, Zhu H Location-aware deep collaborative filtering for service recommendation. Trans Syst I Man Cybernet. https://doi.org/10.1109/TSMC.2019.2931723 (In press)
19. Zhang Y, Wang K, He Q, Chen F, Deng S, Zheng Z, Yang Y Covering-based web service quality prediction via neighborhood-aware matrix factorization. IEEE Trans Serv Comput. https://doi.org/10.1109/TSC.2019.2891517 (in press)
20. Juang B, Rabiner L (1991) Hidden markov models for speech recognition. Technometrics 33(3):251–272
21. Schuller B, Rigoll G, Lang M (2003) Hidden markov model-based speech emotion recognition. In: Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing: 6-10 April 2003. IEEE, Hong Kong
22. Lee K, Hon H (1989) Speaker-independent phone recognition using hidden markov models. IEEE Trans Acoust Speech Signal Process 37(11):1641–1648
23. Lipton Z, Berkowitz J, Elkan C (2015) A critical review of recurrent neural networks for sequence learning. arXiv preprint arXiv:1506.00019
24. Ren S, He K, Girshick R, Sun J (2017) Faster r-cnn: towards real-time object detection with region proposal networks. IEEE Trans Pattern Anal Mach Intell 39(6):1137–1149
25. Gers F, Schmidhuber J, Cummins F (2000) Learning to forget: Continual prediction with lstm. Neural Computation 12(10):2451–2471
26. Kolen J, Kremer S (2001) A Field Guide to Dynamical Recurrent Networks. Wiley-IEEE Press, New York
27. Graves A, Jaitly N (2014) Towards end-to-end speech recognition with recurrent neural networks. In: Proceedings of International Conference on Machine Learning: 3-6 Dec. 2014. Journal of Machine Learning Research (JMLR), Detroit
28. Bin Y, Yang Y, Shen F, Xie N, Shen H, Li X (2018) Describing video with attention-based bidirectional lstm. IEEE Trans Cybern 49(7):2631–2641
29. Dowlin N, Gilad-Bachrach R, Laine K, Lauter K, Naehring M, Wernsing J (2016) Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In: Proceedings of International Conference on Machine Learning: 19-24 June 2016. Journal of Machine Learning Research (JMLR), New York City
30. Liu J, Juuti M, Lu Y, Asokan N (2017) Oblivious neural network predictions via minionn transformations. In: Proceedings of ACM SIGSAC Conference Computer and Communications Security: Oct. 30-Nov. 03 2017. ACM, Dallas
31. Zhang Q, Yang L, Chen Z (2015) Privacy preserving deep computation model on cloud for big data feature learning. IEEE Trans Comput 65(5):1351–1362
32. Yilmaz E, Ferhatosmanoglu H, Ayday E, Aksoy R (2019) Privacy-preserving aggregate queries for optimal location selection. IEEE Trans Dependable Secure Comput 16(2):329–343
33. Huang K, Liu X, Fu S, Guo D, Xu M A lightweight privacy-preserving cnn feature extraction framework for mobile sensing. IEEE Trans Dependable Secure Comput. https://doi.org/10.1109/TDSC.2019.2913362 (In press)

Publisher's Note