Towards Practical Secure Automatic Speech Recognition

Jun-jie Zhou[1] Wen-jie Lu[1] Jun Sakuma[1,2,3]

Abstract: Smart speakers are become popular thanks to the improvements of automatic speech recognition systems. However, the always-on system raises privacy problems. Moreover, we imagine in the future that the service provider may change the smart speakers to be active that the speakers will offer user-interested/user-aimed information according to the user conversations without being firstly evoked, which brings the further privacy concerns. In this study, we address this privacy issue and discover the feasibility of realizing such active smart speakers. Specifically, we mainly investigated the computation cost of using the state-of-the-art multiplication triples generation protocols to prepare for three different speech recognition models and give detailed results. The results show that the DNN model is the most practical secure ASR model at present with existing protocols because it only needs 6.35 hours for preparing private evaluation in the remaining 17.65 hours of a day.

Keywords: Secure Two-party Computation, Automatic Speech Recognition, Deep Neural Network

1. Introduction

Recently, the development of deep learning has succeeded to improve the accuracy of automatic speech recognition (ASR) to be better than human performance [1]. As a result, smart speakers with speech-based AI assistants such as Google Home and Amazon Echo are becoming more natural in communication and thus being so popular that smart speaker owners spent nearly 2.8 hours per day on average [2]. But due to the limitation of the hardware and network, these devices are now playing a passive role that they will not provide services (or information) until locally detected the keywords such as "Ok Google" or "Alexa".

On the other hand, considering about the history of the search engine, we can see a trend that service providers (SP) may change their smart speakers to become active in which situation the speakers will offer information that you need, or they want to advertise according to your speech. That is, smart speakers will do the full speech recognition all the day with the help of the server, and provide the information to users according to the contents or keywords in the speech. The keywords can be both set by service providers for advertising and by users for interests. For example, when you are discussing your trip to Tokyo with friends, and then the speaker asks you whether you want to know more about Tokyo. The speaker will then tell you the weather and the major tourist attraction if you answered yes. However, no matter whether the smart speakers send the speech data to the server for recognition or not, an always-on recording system will cause privacy concerns, such as sending conversations to other users without permission [3]. Furthermore, it raises more concerns when continuously transmitting audio speech to the server and recognize the full speech. Not only users do not want to be recorded all over time and leak to SP but also the SP wants to protect their learning models.

Secure two-party computation (S2C) is a potential solution for such a privacy-preserving machine learning process which allows two players to jointly compute a function on their secret inputs without leaking anything except the result. There are many frameworks proposed for neural networks [4], [5], [6] and they all evaluate the product of matrices based on additive secret sharing with precomputed multiplication triplets (MTs) [7]. If pre-processing is allowed, this method can be very efficient compared with the garbled circuit or homomorphic encryption since it requires no cryptographic operations at processing time and the MTs can be generated independently with inputs as pre-processing. However, we find it still takes a lot of time when used for large-scale computation. For instance, it takes about 9 seconds to evaluate a 28×28 image with a simple CNN model, and nearly 4 seconds are used for generating the MTs [5].

We address the privacy issue and discover the feasibility of realizing the active smart speakers. Individually, we consider two scenarios shown in **Fig. 1**. In both cases, the speaker will record user conversations all the time and encrypt the real-time speech data before sending to the server. The server will conduct the private speech recognition and send the encrypted results back to the speaker. At last, the speaker asks the user whether he wants more information

[1] University of Tsukuba
[2] JST CREST
[3] RIKEN AIP Center

if the speech contains the keywords "Tokyo". We consider two use-cases of smart speakers that actively recognize user conversations. In the case A, the server privately recognizes the whole speech into text; in the case B, the server only classifies the audio with keywords defined in advance. Since data can only be decrypted by the local speaker, the server will not learn anything about the raw speech data as well as the inference results in both cases.

To be practical, we set our goal as that the smart speaker can privately recognize speech no less than 2.8 hours in one day, which means that one-second speech has to be evaluated with precomputing of 7.6 seconds provided that the computation at recognition time is sufficiently short. Therefore, the object of this work, our first step of the practical private speech recognition research, is to investigate how long it takes when we use the state-of-the-art cryptographic protocols to implement the trained ASR models. Especially, we measure the generation time of MTs using secure matrix product (SMP) protocol [8] and semi-honest third party based vector dot product (STP-VDP) protocol [6] respectively when used in the bi-directional recurrent neural network [9], a convolution neural network [10], and a deep neural network [11].

Related Work First, to our best knowledge, [12] in 2007 is the first to discuss the privacy-preserving speech recognition. In that work, computation of HMMs is performed by one proposed secure protocol for computing logsum and several existing cryptographic primitives based on homomorphic encryption(HE) scheme. Further, these techniques are then extended to be used in [13] and [14]. Recently, [15] pointed out that the whole encrypted inputs may be recovered caused by the logsum protocol proposed by [12] and developed a provably secure HMM and GMM computation. However, HMM computation for 3 hidden states and 32 observations (nearly 1 second) in two-party setting costs SP roughly 10 mins and Client about 2.5 mins offline. This is far from our target value 7.6 seconds.

Also, these works are all based on traditional speech systems requiring high-leveled skills to tune the input features and models. The introduction of recent end-to-end deep learning models has improved both convenience and performance [1]. As a natural result of the importance of privacy, there raises a great demand for privacy-preserving deep learning frameworks.

[4] proposed SecureML framework which uses new cryptography-friendly activation functions to securely train the models. They also proposed an OT-based vectorized MTs and Additive HE (AHE) based method to compute inner products of two additively shared vectors. It costs 4.7 seconds to generate for a 2-layer network with 128 neurons each layer.

In 2017, [5] proposed MiniONN framework, using approximated activation functions to securely estimate any given neural network. They used somewhat homomorphic encryption (SwHE) to improve the AHE-based MT generation method. For the same model as in [4], it only takes 3.58 seconds.

Though these costs are within the target value, the model is designed for a image rather than an audio, thus it is too simple to be used for an ASR task. Further, the authors in [8] proposed a much faster MT generation protocol SMP using packing techniques and yields 20-108 times faster than these existing methods [4], [5]. To generate MTs for product of two $[128 \times 128]$ matricies, SMP takes 2.07 seconds while OT-based method [4] takes 10.5 seconds and SwHE-based [5] method takes 125 seconds.

The authors of Chameleon framework [6], different with former two-party-only settings, proposed a way to generate MTs with the help of a Semi-honest Third Party (STP) and optimized it to be only involved in the offline phase. With the participants of the third party, both parties can avoid performing the heavy operations on ciphertexts, eventually shortening the generation time of MTs. It costs 1.34 seconds for generation in the same model with [4], [5].

Therefore, we choose the SMP protocol [8] in the two-party-only setting and STP-VDP protocol in a third-party-allowable setting to investigate the feasibility of the practicle secure ASR tasks.

Contributions In brief, we summarize our contributions as follows:

- We investigated the feasibility of using the state-of-the-art cryptographic protocols to implement a practical speech recognition model and give detailed performance results.
- We experimented and estimated the generation time of MTs when to use two state-of-the-art MT generation protocols in the implementation of existing speech recognition models[10], [17].
- We explored a deep neural network based keyword spotting system which only needs 7.2×10^4 multiplies while yielding an 84.95% accuracy. We verify that this model is considered as a practical secure ASR model at present with existing protocols as it can prepare MTs for 6.35 hours private evaluation in the remaining 17.65 hours.

2. Preliminaries

We first use **Table 1** to introduce notations used in this paper.

2.1 Secure Two-Party Setting and Security

Our work focus on secure two-party computation (S2C) which allows two parties to compute a function without learning each other's inputs jointly. This guarantees the same level of security with running a trusted third party (TTP) and can against the semi-honest adversary who follows the protocol specification but tries to learn redundant information via communication. Precisely, we follow the standard real-world/ideal-world paradigm [18] that the view of a semi-honest adversary is indistinguishable in real-world to that in ideal-world. Also, we assume that either SP or Client is corrupted but not both of them at the same time.

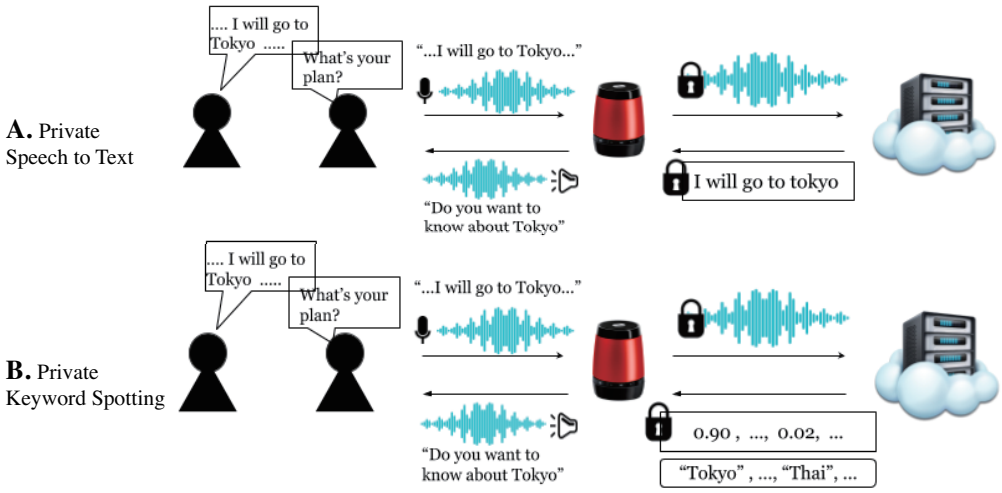

Fig. 1 Two different cases of an active smart speaker processing with the encrypted audio data. In the case A, the server privately recognizes the speech into text while in the case B, the server detects the keywords privately.

Table 1 Notation table

Notation	Meaning
$[n]$	Set of positive integers $\{0, 1, \ldots, n-1\}$
$\boldsymbol{v}, \boldsymbol{v}_i$	Row vector and its i-th entry
$\boldsymbol{M}, \boldsymbol{M}_{i,j}$	Matrices and its (i,j) entry
$\boldsymbol{M}_{[i:]}, \boldsymbol{M}_{[:j]}$	The i-th row and the j-th column of $\boldsymbol{M}$
$P, P[i]$	Polynomial P and its i-th coefficient
X	Indeterminate of polynomials
$e \xleftarrow{\$} \mathcal{E}$	Uniform random sample e from set $\mathcal{E}$
$\langle a \rangle_0, \langle a \rangle_1$	additive share of a
$\mathbb{A}_p$	Ring defined as $\mathbb{Z}_p[X]/(X^m+1)$
p, m	A prime number and a 2-power number
$\langle u, v, z \rangle$	A multiplication triple
L	Length of the input speech
l	Length of a overlapped frame of speech
s_l	Stride size of a overlapped frame of speech
T	Number of speech frames, $T = \frac{L-l+s_l}{s_l}$
F	Number of features extracted from speech
ℓ	The ℓ-th layer in neural network, $\ell \in [d]$
$\boldsymbol{x}^\ell$	Input of the ℓ-th layer, $\boldsymbol{x}^\ell \in \mathbb{R}^{n_1^\ell}$
$\boldsymbol{W}^\ell$	Weight matrix of the ℓ-th layer, $\boldsymbol{W}^\ell \in \mathbb{R}^{n_2^\ell \times n_1^\ell}$
c	Number of lables
$\boldsymbol{K}, n_k$	n_k kernels in convolution, $\boldsymbol{K} \in \mathcal{R}^{n_t^k \times n_f^k}$
s_t, s_f	Convolution stride size in time and frequency

2.2 Semi-honest Third Party

The Semi-honest Third Party (STP) is introduced in [6] to efficiently precompute the correlated randomness in the offline stages. STP can be implemented in three ways: 1) use a trusted hardware 2) use trusted execution environments 3) use a separate computing node. With the help of STP, both parties can generate their correlated randomness without expensive cryptographic operations and thus reducing both computation and communication costs.

2.3 Ring-based Homomorphic Encryption

Beside the STP, we also generate MTs by using a ring-learning with errors [19] of homomorphic encryption (HE) whose message space is defined by the polynomial ring $\mathbb{A}_p := \mathbb{Z}_p[X]/(X^m+1)$ where m is a 2-power number and p is a prime number.

Let $(\mathsf{sk}, \mathsf{pk})$ be a private-public key pair. We write $[\![\cdot]\!]$ to denote the encryption function, and $\mathsf{Dec}_{\mathsf{sk}}(\cdot)$ for the decryption. For elements $A, B \in \mathbb{A}_p$, we leverage the following properties of HE in our construction.

- Additive homomorphism:

$$\mathsf{Dec}_{\mathsf{sk}}([\![A]\!] \oplus [\![B]\!]) = A + B$$
$$\mathsf{Dec}_{\mathsf{sk}}([\![A]\!] \oplus B) = A + B$$

- Multiplication with plain polynomials:

$$\mathsf{Dec}_{\mathsf{sk}}([\![A]\!] \otimes B) = A \times B$$

Notice that the polynomial computations $+$ and $\times$ are done under the ring $\mathbb{A}_p$. The operators $\oplus$ and $\otimes$ respectively indicate homomorphic addition and homomorphic multiplication. Also, we write $\ominus$ to denote the homomorphic subtraction. RLWE-based HE schemes enable us to encrypt various messages in one single ciphertext and process parallelly in the SIMD manner. Typically, we use HElib [20], the most functioning implementation of the symmetric version of the BGV's leveled HE scheme [21].

2.4 Additive Secret Shares

In this paper, we assume that a secret value is shared additively between two parties. That is, when $\textsc{Client}(\mathcal{P}_0)$ and $\mathrm{SP}(\mathcal{P}_1)$ privately distribute a value $a \in \mathbb{Z}_p$ into $\langle a \rangle_0$ and $\langle a \rangle_1 \in \mathbb{Z}_p$, then $a = \langle a \rangle_0 + \langle a \rangle_1 \bmod p$ where p is a prime number. Additionally, let $\langle \boldsymbol{v} \rangle_k$, $\langle \boldsymbol{M} \rangle_k$ denote the additive shares of a vector $\boldsymbol{v}$ and a matrix $\boldsymbol{M}$ for $k \in \{0, 1\}$.

We can perform the addition of two secret shares of x and y without communication. To be specific, $\mathcal{P}_k$ holds $\langle x \rangle_k$, $\langle y \rangle_k \in \mathbb{Z}_p$ and the addition can be realized as $\mathcal{P}_k$ locally conducts the $\langle x \rangle_k + \langle y \rangle_k \bmod p$. However, it is required one round of communication to multiply two secret values.

Multiplication Triples To conduct the multiplication, both parties need to have shares of Beaver's Multiplication Triples (MT) [7], which refers to a triple $\langle \langle u \rangle_k, \langle v \rangle_k, \langle z \rangle_k \rangle$ such that $z = \langle z \rangle_0 + \langle z \rangle_1 = uv = (\langle u \rangle_0 + \langle u \rangle_1)(\langle v \rangle_0 +$

$\langle v \rangle_1$) mod p. It is worthy to mention that the generation of MT is independent on the private inputs from two parties and thus can be prepared during the offline precomputation stage. In the online stage, two parties perform the multiplication of x and y with using the triple. The computation process is simple. $\mathcal{P}_k$ computes $\langle e \rangle_k = \langle x \rangle_k - \langle u \rangle_k$ and $\langle f \rangle_k = \langle y \rangle_k - \langle v \rangle_k$. Then they exchange their shares of $\langle e \rangle_k, \langle f \rangle_k$ to reconstruct e, f. Lastly, $\mathcal{P}_k$ computes $\langle xy \rangle_k = f \times \langle x \rangle_k + e \times \langle y \rangle_k + \langle z \rangle_k + k \times e \times f$.

Typically, we generate MTs following the HE-based Secure Matrix Product (SMP) protocol [8] in this work. The authors of SMP use the double-packing technique consist of Forward-backward encoding [22] and the Chinese-Remainder-Theorem packing [23] to efficiently encode messages, and it allows to batch several inner products with only one single homomorphic multiplication, which accelerates the generation of MTs. However, both two parties still have to conduct heavy cryptographic operations, which can be avoided by introducing an STP as [6]. We refer readers to [6], [8] for more details.

3. Fundamental Deep Learning Functionalities

Many different models, such as simple deep neural networks [11], [17], [24] only consists of fully connected layers, convolutional neural networks [10], [25] and more complicated models like RNNs [9], can be applied to the task of speech recognition. Though there is a variety of models, they share some similar fundamental functionalities. We introduce these foundations as summarized in the **Table 2**.

In a d-layer nueral network, we denote the input and the weight matrix in the ℓ-th layer as $\boldsymbol{x}^{(\ell)} \in \mathbb{R}^{n_1^\ell}$ and $\boldsymbol{W}^\ell \in \mathbb{R}^{n_2^\ell \times n_1^\ell}$, where $\ell \in [d]$. Especially, we denote the input of the network as $\boldsymbol{x}^{(0)}$.

Table 2 Basic functionalities of different layers

Layer	Functionality
FC	$\mathsf{FC}(\boldsymbol{W}^\ell, \boldsymbol{x}^{(\ell)}) = \boldsymbol{W}^\ell \boldsymbol{x}^{(\ell)}$
Conv	$\mathsf{Conv}(\boldsymbol{K}^\ell, \boldsymbol{T}^\ell)_{i',j'} = \sum_{i=0}^{n_t^k-1} \sum_{j=0}^{n_f^k-1} \boldsymbol{K}_{i,j}^\ell \cdot \boldsymbol{T}_{i's_t+i,j's_f+j}^\ell$
BiR	$\mathsf{BiR}(\boldsymbol{W}^\ell, \boldsymbol{x}^{(\ell)}) = g(\boldsymbol{W}^\ell \boldsymbol{x}^{(\ell)} + \boldsymbol{W}_r \boldsymbol{x}^{(\ell)})$
ReLU	$\mathsf{ReLU}(x) = \max(0, x)$

3.1 Linear Functions

Fully Connected Among all the models, the most fundamental and commonly used layer is the fully connected layer, which connects every node that consists of the previous layer to every node in the present layer. We can evaluate the ℓ-th fully connected layer by conducting a matrix to vector production in formula

$$\mathsf{FC}(\boldsymbol{W}^\ell, \boldsymbol{x}^{(\ell)}) = \boldsymbol{W}^\ell \boldsymbol{x}^{(\ell)} \tag{1}$$

yielding $N_{FC} = n_1^\ell n_2^\ell$ multiplications.

Convolution Next, we introduce the convolutional layer, which conducts a convolution operation taking advantage of the two-dimensional inputs such as images and speech signals. If the ℓ-th layer is the convolutional layer, then we denote the input as $\boldsymbol{T}^\ell \in \mathbb{R}^{n_t^\ell \times n_f^\ell}$. There is a multiplication window called kernel, denoted as $\boldsymbol{K} \in \mathbb{R}^{n_t^k \times n_f^k}$, and there can be n_k kernels in the layer. The kernel is stridden by (s_t, s_f) on the input matrix to compute the output where s_t, s_f are referred to as convolution stride size in time and frequency respectively. We write such convolution operation as:

$$\mathsf{Conv}(\boldsymbol{T}^\ell, \boldsymbol{K}^\ell)_{i',j'} = \sum_{i=0}^{n_t^k-1} \sum_{j=0}^{n_f^k-1} \boldsymbol{K}_{i,j}^\ell \cdot \boldsymbol{T}_{i's_t+i,j's_f+j}^\ell \tag{2}$$

yielding $N_{conv} = \frac{(n_t^k n_f^k)(n_t^\ell - n_t^k + s_t)(n_f^\ell - n_f^k + s_f)}{s_t s_f}$ multiplications each kernel and $N_{Conv} = n_k N_{conv}$ multiplications in total.

Bi-directional Recurrent A bi-directional recurrent layer is used in [9], including a set of forward recurrence and a set of backward recurrence. Except the ℓ-th weight matrix $\boldsymbol{W}^\ell$, there also exists one forward weight matrix $\boldsymbol{W}_r^f$ and one backward weight matrix $\boldsymbol{W}_r^b$ has the same size as $\boldsymbol{W}^\ell$. This layer is usually used in the model for speech processing. Denote the length of the raw speech input to be L and we write each time-sliced input as $\boldsymbol{x}_t^{(0)}$ where $t \in [L]$. Then the ℓ-th layer's input become $\boldsymbol{x}_t^{(\ell)}$ and the $\mathsf{BiR}(\boldsymbol{W}^\ell, \boldsymbol{x}_t^{(\ell)})$ operation can be written as

$$\mathsf{BiR}_f(\boldsymbol{x}_t^{(\ell)}) = g(\boldsymbol{W}^\ell \boldsymbol{x}_t^{(\ell)} + \boldsymbol{W}_r^f \boldsymbol{x}_{t-1}^{(\ell)}), \tag{3}$$

$$\mathsf{BiR}_b(\boldsymbol{x}_t^{(\ell)}) = g(\boldsymbol{W}^\ell \boldsymbol{x}_t^{(\ell)} + \boldsymbol{W}_r^b \boldsymbol{x}_{t+1}^{(\ell)}), \tag{4}$$

where $g()$ represents some activation function. In the forward recurrence, $\boldsymbol{x}_t^{(\ell)}$ should be computed from $t = 0$ to $t = L - 1$, and should be computed from $t = L - 1$ to $t = 0$ in the backward recurrence. Eventually, it totally computes $2Ln_1^\ell n_2^\ell$ multiplications. For the convenice, we simplify $\mathsf{BiR}(\boldsymbol{x}_t^{(\ell)})$ to be $\mathsf{BiR}(\boldsymbol{x}^{(\ell)}) = g(\boldsymbol{W}^\ell \boldsymbol{x}^{(\ell)} + \boldsymbol{W}_r \boldsymbol{x}^{(\ell)})$.

We can learn from Table 2 that not only different models share some common functionalities but also the linear functionalities have similar operations. That is, all of them can be computed by matrix-vector products except a few nonlinear operations, which requires MTs for multiplications. Hence, we can calculate the fully connected layer, convolution layer and bi-directional recurrent layer in the secure domain if we have precomputed a sufficiently large number of MTs.

3.2 Activation Functions

Rectified Linear Unit A rectified Linear unit (ReLU) is usually applied after the fully connected layer, taking the positive part of its input.

$$\mathsf{ReLU}(x) = \max(0, x)$$

where x is a real value. It can be trivial to implement

$\mathsf{ReLU}(x)$ in our secure two-party settings by using ABY library[26] as follows:

$$\mathsf{ReLU}(x) = compare(x, 0) \cdot x, \tag{5}$$

where $compare(x, 0) = 1$ if $x \geq 0$ else 0. The operations we need, i.e., $+, -, \times$ and *compare* (CMP) are already supported by the ABY library.

Softmax The softmax function is usually used in the output layer, calculating the probabilities of each target class over all possible target classes. Typically, assuming that we have k classes, the softmax takes the input $\boldsymbol{x} \in \mathbb{R}^k$ and outputs a real vector $\tilde{\boldsymbol{x}} \in \mathbb{R}^k$ where $\tilde{x}_i = e^{x_i} / \sum_j e^{x_j}$ for $i \in [k]$. Considering that the softmax is applied only in the output layer and what can be inferred from $\tilde{\boldsymbol{x}}$ is the same as what can be inferred from $\boldsymbol{x}$, we argue that there is no necessity to implement the softmax in the domain of secure computation. As a result, the softmax function is evaluated as the post process with the plaintext of the output in this work.

4. Automatic Speech Recognition

The automatic speech recognition (ASR) is a process that converts a speech signal into a sequence of words. Thanks to the big data and the development of deep learning, the performance of ASR has been significantly improved [9], [27]. We consider two types of ASR tasks that can be used in the smart speakers: Speech to text task and keyword spotting task. Both of them can recognize contents contained in the speech while the difference between them is that a speech to text task has the ability to recognize the full speech into text, but a keyword spotting task can only detect the keywords in the speech.

We will present the process of both tasks as well as specific existing models designed for them in short.

4.1 Feature Extraction

Before we conduct the speech recognition task, we have to first preprocess the data, i.e., extract features from the raw audio signal. The most common technique is the Mel-frequency cepstral coefficients (MFCC) as it is known to be effective and robust under various conditions [28].

Fig. 2(a) shows a one-second speech signal processing when uttering "cat". We set the total length of the signal to be L and each overlapping frames' length to be l with a stride s. This gives us $T = \frac{L-l+s}{s}$ frames in total. The original speech signal is in an extremely high dimension that is hard to deal with. To compress the dimension, we then extract F features from each frame, generating a feature map of size $T \times F$ for the entire speech signal shown in the Fig. 2(b). In this work, we also take MFCCs as the speech features. The raw speech signal in the time domain is eventually translated into a set of spectral coefficients in the frequency domain, which is much lower in dimension and can be handled much easier.

Even though a larger feature map will result in a better performance as more features of the speech signal will be taken into account, a considerably large input will increase the multiplications and raise the generation time of MTs in the end.

4.2 End-to-end Speech-to-Text

A speech-to-text system aims at recognizing the audio signal and change the audio into its corresponding text. It can be applied to many situations such as speech translation, caption generation and voice operations of IoT home appliances. Especially, recent smart speakers can understand the users' commands by voice and response to the commands.

Fig. 2(A) shows a process of speech to text task where the input is MFCC features extracted from an audio signal and output the text. We concentrate on the DeepSpeech [9] model which uses BiRNN as the recognition model.

Bi-directional Recurrent Neural Network The BiRNN model, a state-of-the-art ASR model, is introduced in the DeepSpeech [9], yielding a 16% word error rate over Hub5'00 dataset. We use the trained model implemented by [29] for our estimation. The raw input, a nearly one-second audio signal, is extracted into a $[19 \times 26]$ feature map, which is flattened to a vector and fed as the input of the network. Output is a probability sequence of labels including 26 characters, *space*, *apostrophe* and *blank*. This BiRNN model is composed of 5 layers of hidden units. The first three and the fifth layers are fully connected layers followed by a clipped rectified-linear activation function in each layer. The fourth layer is a bi-directional recurrent layer including one set with forward recurrence and one set with backward recurrence. The output layer is the standard softmax function.

4.3 Keyword Spotting

Keyword Spotting (KWS) is a task that aims at isolating and detecting predefined keywords in a speech signal, which can be regarded as a special case of speech recognition. It is a core technology in the interaction between speech based AI system and users by evoking the device with voice especially for devices with limited computing resources such as smartphones and smart speakers. As long as the keywords are detected, the device will be activated and provide further interaction with users with the help of powerful servers. For good user experience, KWS is preferred to be highly accurate and low latency in detecting the words, which makes it necessary for KWS to be always-on and react in real-time.

As shown in Fig. 2(B), the KWS can be viewed as a classification problem over the audio signal. We mainly investigated convolution neural network (CNN) and deep neural network (DNN) models to be the classifier of KWS. For both models of KWS, 40 MFCC features are extracted from speech frames of length $l = 40[ms]$ with stride $s_l = 20[ms]$. This setting leads to the input feature map as 49×40 from a one-second audio. In the final of the network, a vector of probabilities of 12 predefined keywords is outputted.

Convolution Neural Network CNNs [30] become popular in the filed of acoustic modeling because of the performance improments over the DNNs [10] and less complex

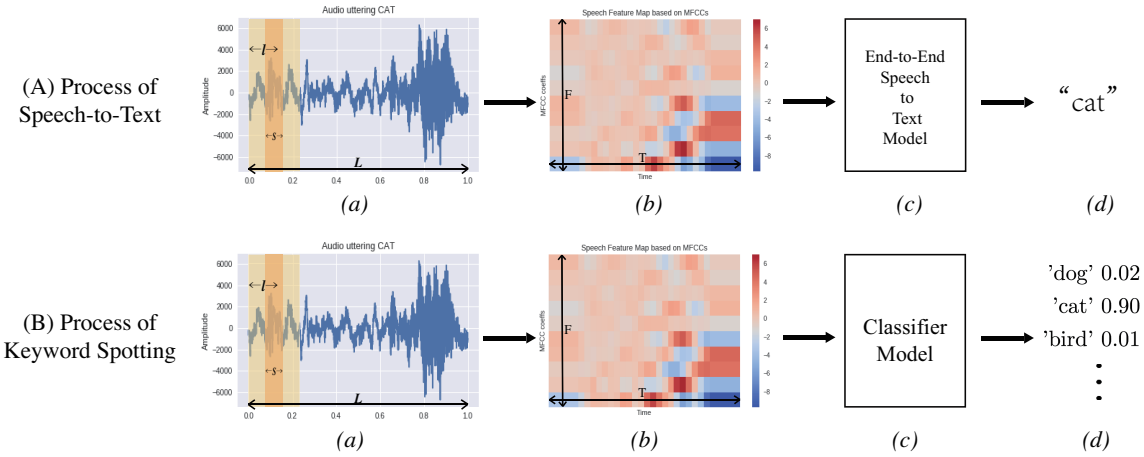

Fig. 2 Image of two different ASR tasks with the same audio input uttering "cat": (A) shows the process of a Speech-to-Text task where the output should be the text "cat"; (B) indicates the process of a Keyword Spotting task where the outputs is a vector consists of probabilities on each label.

than the RNNs. The trained model `cnn-one-fstride4` we use is proposed in [10] and implemented in [17]. The trained model yields about 84.6% test accuracy over the Google speech commands dataset [31].

`cnn-one-fstride4` is referred to this name as it only strides the kernel in frequency by $s_f^k = 4$. It has 186 kernels of size $[49 \times 8]$ in all. Such a setting let the network be more sensitive to the frequency than the time. Next to the convolution layer is two fully connected layers and the output layer.

Deep Neural Network DNN is a standard feed-forward neural network model consists of d fully-connected layers of h hidden units. The ℓ-th hidden layer holds a weight matrix $\boldsymbol{W}^\ell$, and an activation function follows each hidden layer. In detail, the flattened frequency-time feature matrix $\boldsymbol{x}^0 \in \mathbb{R}^{TF}$ is fed as input. The first $d-1$ layers are followed by the ReLU function and the last output layer employs the softmax as the activation function, giving the probabilities of $c = 12$ predefined keywords.

We use the trained model implemented in [17], which is a 4-layer DNN with 144 units every layer. We denote this model as `dnn-mf40`. The test accuracy is about 84.3%.

5. Evaluations

We can see from the former studies [4], [5] that the precomputing time for a neural network is much longer than the online evaluation time. Hence, in this work, we conduct the comparisons mainly on the generation time of MTs between three different models mentioned in the former section using two distinct MT generation protocols. To be more precise, we measured and estimated the end-to-end computing time when running the SMP protocol [8] and the STP based Vector Dot Product STP-VDP protocol [6] at each layer, respectively.

We experimented the SMP protocol in C++ and compiled with gcc-6.3 on Ubuntu 14.04. We use HElib [20] where the parameters $m = 4096$ and $p = 70913$ were used to provide 128 slots just as in [8] and provide at least 80-bit security level. We ran experiments on two machines (Xeon E5-2640 v3@2.60 GHz CUP, 32 GB RAM, GeForce GTX 1080 $\times$ 1 and Quadro K2200 $\times$ 1) within a LAN with less than 0.1 ms ping delay.

5.1 Comparison Results

The comparison results of two protocols applying to Deepspeech [9], `cnn-one-fstride4` [10] and `dnn-mf40` [17] along with the architectures of models are given in **Table 3**, **Table 4** and **Table 5** respectively.

We can find in Table 3 that all the weight matrices in the Deepspeech model are larger than those of the CNN and the DNN models, which indicates that the speech-to-text task is much more complicated than the keyword spotting task. Also, the BiRNN layer computing with time series forward and backward leads enormous multiplications and increases computation costs. This result is so time-consuming that it costs at least 77 minutes to precompute the MTs. Our evaluation shows that even a whole day (24 hours) is used for preparing MTs, only about 18 seconds can be evaluated privately through the Deepspeech model as confirmed by our results shown in Table 3.

On the other hand, the number of multiplications is reduced 99.97%-99.99% when changing from a speech-to-text task to a keyword spotting task. As the Table 4 indicates, both methods cost most of the time for the convolutional operation. Notably, the SMP protocol costs 60 times more than the STP-VDP protocol in the convolution layer. The reason for this is because that the SMP protocol has to run $n_k = 186$ times for each kernel while STP-VDP protocol can send the random seeds in advance. Besides, 2.96 seconds is within the required 7.6 seconds while the SMP is still far from it.

Though the accuracy of DNN is a little worse than the CNN model, it only needs 2.76 seconds using SMP and 0.14 seconds using STP-VDP to generate MTs due to the small

size of the model. Both of the methods can fulfill the requirement of time as well as having an acceptable test accuracy. Hence, we can conclude that considering our goal of using the private privacy-preserving smart speakers at least 2.8 hours one day, the DNN model is the most practical solution among the others.

Table 3 BiRNN (Deepspech) model for Speech-to-Text task and the comparison of MTs generataion time between running SMP [8] and estimating from [6]. We averaged the performance of SMP from 50 runs and estimated the performance of STP-VDP by calculating the number of multiplications per layer.

Layer	Weight Size	Time [s] by SMP	Time [s] by STP-VDP
FC	$[2048 \times 494]$	5.25	1.77
FC	$[2048 \times 2048]$	15.34	7.34
FC	$[4096 \times 2048]$	31.10	14.67
BiR	$[8192 \times 6144] \times 2 \times 26$	9571.74	4577.56
FC	$[2048 \times 4096]$	29.52	14.67
FC	$[27 \times 2048]$	0.80	0.10
In total	2.6×10^9 Mults	9653.65	4616.11

Table 4 `cnn-one-fstride4` model for keyword spotting task and the comparison of MTs generataion time between running SMP [8] and estimating from [6].

Layer	Weight Size	Time [s] by SMP	Time [s] by STP-VDP
Conv(kernel)	$[8 \times 98]$ count = 1674	122.76	2.55
FC	$[128 \times 1674]$	0.84	0.91
FC	$[128 \times 128]$	0.65	0.07
FC	$[12 \times 128]$	0.63	0.01
In total	767872 Mults	124.88	3.54

Table 5 DNN model for keyword spotting task and the comparison of MTs generataion time between running SMP [8] and estimating from [6].

Layer	Weight Size	Time [s] by SMP	Time [s] by STP-VDP
FC	$[144 \times 250]$	0.69	0.15
FC	$[144 \times 144]$	0.67	0.09
FC	$[144 \times 144]$	0.67	0.09
FC	$[12 \times 144]$	0.63	0.01
In total	79200 Mults	2.76	0.34

5.2 Exploration of DNN

We further explored the DNN models such as changing the parameters in the feature extraction phase to seek a smaller size model with better accuracy. We trained the DNN model with the TensorFlow [32] using the Google speech commands dataset [31], which includes 65K audio samples of a length of one-second each. We chose ten keywords [*1] along with "silence" and "unknown" 12 labels in total. We also tuned the parameters in the preprocessing phase to reduce the size of the feature map while keeping the basic architecture as in [17].

The speech is sliced with $l = 120[ms]$ and a stride length of $s_l = 55[ms]$, and this brings $T = \frac{L-l+s_l}{s_l} = 17$ frames. Each frame has 12 features and turnouts the input size being $T \times F = 17 \times 12$, the key to the decrement of multiplies. In summary, the trained DNN composed of $d = 4$ layers and each layer is the same size as Table 5 except that the size of the first layer is 144×204. The active functions of the first three layers are ReLU, and the last layer is softmax. We denote this model as `dnn-mf12`.

Experiment results in **Table 6** shows that our `dnn-mf12` achieved a test accuracy of 84.95%, which is better than `cnn-one-fstride4`. Also, it reduces around 8% multiplies compared with [17].

We measured the end-to-end time and communication cost of running SMP and estimated the time and communication cost using data in [6]. The result is summarized in 6. Since the size changed slightly, the generation times are almost the same as the `dnn-mf40`. Though the STP-VDP is faster than SMP, thanks to the packing techniques, SMP costs 3 times less than the STP-VDP in communication.

6. Conclution and Future Work

In this work, we investigated the feasibility of a practical privacy-preserving speech recognition system under the secure two-party computation setting. In detail, we experimentally measured the generation time of multiplication triples by SMP protocol and compared with the estimation time using STP-VDP protocol applied to three different existing ASR models. We find that STP-VDP protocol can make MTs more efficient than SMP protocol and the DNN model is the most realistic model to be implemented in the secure domain.

Next, we tune the parameters such as the length of framed audio, number of features, and train the DNN models in cleartext. We obtain a better model than [17] which yields about 84.95% test accuracy and only requires 72K multiplications. It only takes 0.33-2.76 seconds and 276.1-798.6 KB communication costs to prepare the secure evaluation of one-second audio at pre-processing phase, which means that MTs for 6.35 hours private evaluation can be generated with the remaining time of the day (17.65 hours).

Due to the enormous number of parameters, only the most basic DNN model which ignores the local temporal and spectral correlation in the input speech features[17] can be practically used with the state-of-the-art cryptographic protocols. Even the CNN takes too long time during the precomputing phase, not to mention the time series model such as BiRNN model. The CNN model costs most of the time in the convolution layer blaming to the number of kernels n_k. The number of times that protocols have to be run will grow with the n_k. On the other side, the BiRNN model is significantly large in the size of the weight matrices and the bi-direction layer is necessary to be computed on a time series. These are the two primary reasons causing high latency.

[*1] "Yes", "No", "Up", "Down", "Left", "Right", "On", "Off", "Stop", "Go"

Table 6 Comparing generation time and communication cost of SMP [8] and STP-VDP [6] applying to our trained `dnn-mf12`.

Weight Size	Time [s]		Communication [KB]	
	by SMP	by STP-VDP	by SMP	by STP-VDP
$[144 \times 204]$	0.69	0.14	118.3	323.4
$[144 \times 144]$	0.67	0.09	75.5	228.1
$[144 \times 144]$	0.67	0.09	75.5	228.1
$[12 \times 144]$	0.63	0.01	6.8	19.0
72576 Mults	2.76	0.33	276.1	798.6

Thereby, our next step is to consider more efficient protocols for faster generating MTs especially for the use of convolution and bi-directional layers.

Acknowledgment. This work is supported by JST CREST JPMJCR1302 and Scientific Research Funding 16H02864.

References

[1] Xiong, W., Wu, L., Droppo, J., Huang, X. and Stolcke, A.: The Microsoft 2017 Conversational Speech Recognition System, *Proc. IEEE ICASSP*, IEEE, pp.5934–5938, 2018.

[2] Kinsella, B.: Smart Speaker Owners Use Voice Assistants Nearly 3 Times Per Day, `https://goo.gl/xokxXB`.

[3] Times, T. N. Y.: Is Alexa Listening? Amazon Echo Sent Out Recording of Couple's Conversation, `https://goo.gl/ej8JeR`.

[4] Mohassel, P. and Zhang, Y.: SecureML: A System for Scalable Privacy-Preserving Machine Learning, *2017 IEEE Symposium on Security and Privacy (SP)*, pp.19–38, 2017.

[5] Liu, J., Juuti, M., Lu, Y. and Asokan, N.: Oblivious Neural Network Predictions via MiniONN Transformations, *IACR Cryptology ePrint Archive*, Vol. 2017, p.452, 2017.

[6] Riazi, M. S., Weinert, C., Tkachenko, O., Songhori, E. M., Schneider, T. and Koushanfar, F.: Chameleon: A Hybrid Secure Computation Framework for Machine Learning Applications, *CoRR*, Vol. abs/1801.03239, 2018.

[7] Beaver, D.: Efficient Multiparty Protocols Using Circuit Randomization, *Advances in Cryptology — CRYPTO '91* (Feigenbaum, J., ed.), Berlin, Heidelberg, Springer Berlin Heidelberg, pp.420–432, 1992.

[8] Lu, W. and Sakuma, J.: Faster Multiplication Triplet Generation from Homomorphic Encryption for Practical Privacy-Preserving Machine Learning under a Narrow Bandwidth, *IACR Cryptology ePrint Archive*, Vol. 2018,p.139, 2018.

[9] Hannun, A. Y., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A. and Ng, A. Y.: DeepSpeech: Scaling up end-to-end speech recognition, *CoRR*, Vol. abs/1412.5567, 2014.

[10] Arik, S. O., Kliegl, M., Child, R., Hestness, J., Gibiansky, A., Fougner, C., Prenger, R. and Coates, A.: Convolutional Recurrent Neural Networks for Small-Footprint Keyword Spotting, *ArXiv e-prints*, 2017.

[11] Wang, Z., Li, X. and Zhou, J.: Small-footprint Keyword Spotting Using Deep Neural Network and Connectionist Temporal Classifier, *ArXiv e-prints*, 2017.

[12] Smaragdis, P. and Shashanka, M.: A Framework for Secure Speech Recognition, *Trans. Audio, Speech and Lang. Proc.*, Vol. 15, No. 4, pp. 1404–1413, 2007.

[13] Shashanka, M.: A Privacy Preserving Framework for Gaussian Mixture Models, *2010 IEEE International Conference on Data Mining Workshops*, pp. 499–506, 2010.

[14] Pathak, M. A.: *Privacy-Preserving Machine Learning for Speech Processing*, Springer Publishing Company, Incorporated, 2012.

[15] Aliasgari, M., Blanton, M. and Bayatbabolghani, F.: Secure computation of hidden Markov models and secure floating-point arithmetic in the malicious model, *International Journal of Information Security*, Vol. 16, No. 6, pp.577–601, 2017.

[16] Du, W. and Atallah, M. J.: *Protocols for Secure Remote Database Access with Approximate Matching*, pp.87–111, Springer US, 2001.

[17] Zhang, Y., Suda, N., Lai, L. and Chandra, V.: Hello Edge: Keyword Spotting on Microcontrollers, *ArXiv e-prints*, 2017.

[18] Canetti, R.: Security and Composition of Multiparty Cryptographic Protocols, *Journal of Cryptology*, Vol. 13, No. 1, pp.143–202, 2000.

[19] Lyubashevsky, V., Peikert, C. and Regev, O.: On Ideal Lattices and Learning with Errors over Rings, *Advances in Cryptology – EUROCRYPT 2010* (Gilbert, H., ed.), Berlin, Heidelberg, Springer Berlin Heidelberg, pp.1–23, 2010.

[20] Shai Halevi, V. S.: HELib, `http://shaih.github.io/HElib`.

[21] Brakerski, Z., Gentry, C. and Vaikuntanathan, V.: (Leveled) Fully Homomorphic Encryption Without Bootstrapping, *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, New York, NY, USA, ACM, pp.309–325, 2012.

[22] Yasuda, M., Shimoyama, T., Kogure, J., Yokoyama, K. and Koshiba, T.: Secure Pattern Matching Using Somewhat Homomorphic Encryption, ¥em Proceedings of the 2013 ACM Workshop on Cloud Computing Security Workshop, CCSW '13, New York, NY, USA, ACM, pp.65–76, 2013.

[23] Smart, N. P. and Vercauteren, F.: Fully homomorphic SIMD operations, *Designs, Codes and Cryptography*, Vol. 71, No. 1, pp.57–81, 2014.

[24] Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N. and Kingsbury, B.: Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups, *IEEE Signal Processing Magazine*, Vol. 29, No. 6, pp.82–97, 2012.

[25] Abdel-Hamid, O., Mohamed, A., Jiang, H. and Penn, G.: Applying Convolutional Neural Networks concepts to hybrid NN-HMM model for speech recognition, *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp.4277–4280, 2012.

[26] Demmler, D., Schneider, T. and Zohner, M.: ABY - A Framework for Efficient Mixed-Protocol Secure Two-Party Computation, *22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA, February 8-11, 2015*, 2015.

[27] Saon George, Sercu Tom, R. S. and J., K. H.-K.: The IBM 2016 English Conversational Telephone Speech Recognition System, *arXiv e-prints*, 2016.

[28] Gupta, S., Jaafar, J., Wan Ahmad, W. F. and Bansal, A.: Feature Extraction Using Mfcc, Vol. 4, pp.101–108, 2013.

[29] Mozilla: Mozilla/DeepSpeech, `https://github.com/mozilla/DeepSpeech`.

[30] LeCun, Y. and Bengio, Y.: The Handbook of Brain Theory and Neural Networks, MIT Press, Cambridge, MA, USA, chapter Convolutional Networks for Images, Speech, and Time Series, pp.255–258, 1998.

[31] Warden, P.: Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition, *ArXiv e-prints*, 2018.

[32] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C.,... Zheng, X.: TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015.