Privacy-Preserving Object Detection for Medical Images With Faster R-CNN

Yang Liu, Zhuo Ma, *Member, IEEE*, Ximeng Liu, *Member, IEEE*, Siqi Ma, and Kui Ren, *Fellow, IEEE*

***Abstract*— In this paper, we propose a lightweight privacy-preserving Faster R-CNN framework (SecRCNN) for object detection in medical images. Faster R-CNN is one of the most outstanding deep learning models for object detection. Using SecRCNN, healthcare centers can efficiently complete privacy-preserving computations of Faster R-CNN via the additive secret sharing technique and edge computing. To implement SecRCNN, we design a series of interactive protocols to perform the three stages of Faster R-CNN, namely feature map extraction, region proposal and regression and classification. To improve the efficiency of SecRCNN, we improve the existing secure computation sub-protocols involved in SecRCNN, including division, exponentiation and logarithm. The newly proposed sub-protocols can dramatically reduce the number of messages exchanged during the iterative approximation process based on the coordinate rotation digital computer algorithm. Moreover, the effectiveness, efficiency and security of SecRCNN are demonstrated through comprehensive theoretical analysis and extensive experiments. The experimental findings show that the communication overhead in computing division, logarithm and exponentiation decreases to 36.19%, 73.82% and 43.37%, respectively.**

***Index Terms*— Privacy-preserving, faster R-CNN, medical images, additive secret sharing.**

I. Introduction

FASTER region conventional neural network (R-CNN) based object detection has been considered to be an ideal approach to assist medical diagnosis. Doctors can utilize the automatic detection results of medical images to obtain further insights into the patient-specific pathological features and make a more accurate diagnosis. Some typical applications include anatomical object localization [1], cell tracking [2] and bone age estimation [3]. The detection accuracy of most applications far exceeds that of experienced clinicians. However, the computing power and storage capacity requirements for training such a Faster R-CNN based object detection model are quite staggering. For a single patient, the recording of magnetic resonance imaging (MRI) or computed tomography (CT) required to be processed and stored can be ten minutes or more, and the duration to record data for 1000 patients can reach more than 130 hours [4]. Also, the quality of medical images is higher than the images for general use. Therefore, instead of building their own local server, healthcare centers prefer to outsource their medical images to cloud servers and build the Faster R-CNN model using cloud computing technology. Moreover, to provide a timely response and steady communication for the field diagnostic test, the connection channels with the cloud server should have low communication latency and high robustness against network fluctuations. To achieve this goal, edge computing is proposed to build a "short bridge" between the data owner and the cloud server. Research shows that for deep learning based face recognition, the response latency can be decreased from 900 ms to 169 ms with the help of edge computing [5].

In addition, the high performance of Faster R-CNN based object detection depends heavily on the quality of large-scale training data. A single healthcare center's data are usually not enough to train a high-performance model [6]. Consequently, the cooperation of multiple healthcare centers through data sharing becomes imperative under the situation. Nevertheless, the problem arsing from such collaboration is that the medical images may be leaked to others during the data sharing process. Medical images are considered to be private information of patients. No patient wants these images to be revealed to others, except for the healthcare center where the patient is treated. At the same time, medical images are valuable commercial resources for the healthcare center. Taking the cancer diagnosis as an example, at least two days are required to process only one patient's pathological images with metaiodobenzylguanidine (a common technology for collecting pathological images from patients with cancer).[1] A healthcare center may spend years building its pathological database and cannot be willing to share the data with others. Consequently, for patient privacy and smooth cooperation between healthcare centers, it is necessary to build an efficient

Manuscript received April 2, 2019; revised July 3, 2019, September 9, 2019, and October 3, 2019; accepted October 3, 2019. Date of publication October 11, 2019; date of current version December 17, 2021. This work was supported in part by the National Natural Science Foundation of China Grant U1804263, Grant U1764263, Grant 61872283, and Grant 61702105, in part by the China 111 Project under B16037, in part by the College of Mathematics and Computer Science, Fuzhou University, China, and in part by the Fujian Provincial Key Laboratory of Information Security of Network Systems, Fuzhou, China. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Aris Gkoulalas Divanis. *(Corresponding author: Zhuo Ma.)*

Yang Liu and Zhuo Ma are with the School of Cyber Engineering, Xidian University, Xi'an 710071, China (e-mail: bcds2018@foxmail.com; mazhuo@mail.xidian.edu.cn).

Ximeng Liu is the with College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350108, China (e-mail: snbnix@gmail.com).

Siqi Ma is with Data 61, CSIRO—Marsfield Site, Marsfield, NSW 2122, Australia (e-mail: siqi.ma@csiro.au).

Kui Ren is with the Institute of Cyberspace Research, Zhejiang University, Zhejiang 310027, China (e-mail: kuiren@zju.edu.cn).

Digital Object Identifier 10.1109/TIFS.2019.2946476

[1] https://www.insideradiology.com.au/mibg-scan/

privacy-preserving framework for Faster R-CNN based object detection of medical images.

For medical image privacy, current research mostly concentrates on data storage privacy and cannot support online calculations in encrypted format [7]. The problem of the method is that when applying the medical image data into Faster R-CNN, we still have to query and download the data to a local server, which can dramatically reduce data availability and computational efficiency. To overcome this problem, schemes based on homomorphic encryption (HE) [8] and garbled circuit (GC) [9] have been proposed. However, HE and GC are both computation-intensive and memory-intensive algorithms. For most real-world applications, the overheads caused by these methods are almost intolerable. Additionally, differential privacy (DP) is also a popular technique for the privacy preservation of deep learning models. Implementing DP only requires few computations to generate random perturbations. However, the accuracy reduction caused by the introduction of random perturbations is quite considerable [10].

To address the above problems, we propose an additive secret sharing based Faster R-CNN framework (SecRCNN) for privacy-preserving object detection of medical images. The main contributions of this work are listed as follows.

- We propose SecRCNN, the first privacy-preserving Faster R-CNN framework for medical image object detection. SecRCNN allows multiple healthcare centers to securely share their medical image data and collaborate to build a high-performance Faster R-CNN model to assist in clinical diagnosis. During the cooperation process, no healthcare center has to worry about their own data revealed to other healthcare centers or the cloud server.
- Several additive secret sharing based sub-protocols are designed to complete the division, exponentiation and logarithm operations in SecRCNN. Compared with existing protocols, the newly proposed protocols not only realize the corresponding functions safely and accurately, but also dramatically reduce communication costs.
- A series of interactive protocols are proposed to complete the secure computation of the feature extraction network, the region proposal network and the classification & bounding box regression of Faster R-CNN without revealing the original data.
- A comprehensive analysis is presented to prove the correctness and security of SecRCNN. The experimental findings further indicate that SecRCNN is efficient and only introduces little computation errors on the final output. The decreases of communication overhead for computing division, logarithm and exponentiation reach 36.19%, 73.82% and 43.37%, respectively.

The remainder of this paper is organized as follows. In Section II, we describe the preliminaries of SecRCNN. In Section III, the system model of SecRCNN is described. Then, we introduce the sub-protocols for division, exponentiation and logarithm in Section IV, followed by the interactive protocols utilized to complete the object detection task of SecRCNN in Section V. The security analysis and performance evaluation are presented in Sections VI and VII. Related work is discussed in Section VIII. Section IX concludes this paper.

II. Preliminaries

A. Faster Region Conventional Neural Network

Faster R-CNN is a deep learning architecture that outperforms most previous techniques in object detection and classification [11]. Given a set of medical images I, the first stage of Faster R-CNN is to extract fixed size feature maps m by a common neural network such as VGG-16 [12]. Based on m, the region proposal network (RPN) [11] then produces a predicted bounding box to label the boundaries of the target object and determine its class. The objective loss function of Faster R-CNN corresponds to the addition of the classification loss and the bounding box regression loss, and is given as follows:

$$Loss = \frac{1}{N_c}\sum_{i=0}^{N_c} L_{Log}(p_i, p_i^*) + \lambda \frac{1}{N_r}\sum_{i=0}^{N_r} p_i^* Smooth_{L_1}(t_i, t_i^*), \tag{1}$$

where N_c and N_r are the mini-batch size and the number of anchor locations ; i is the index of anchors; p_i is the prediction probability of whether the anchor i is the target object; p_i^* is the ground-truth label; λ is a constant which is set to 10 by default. Given an input x, smooth L_1 function $Smooth_{L_1}$ works for the bounding box regression loss by computing

$$Smooth_{L_1}(x) = \begin{cases} \dfrac{\sigma^2 x^2}{2}, & |x| < \dfrac{1}{\sigma^2} \\ |x| - \dfrac{1}{2\sigma^2}, & otherwise \end{cases}, \tag{2}$$

where t_i and t_i^* are the coordinates of the predicted bounding box and the ground-truth bounding box, respectively. Let us denote the four types of coordinates as x, y, w and h. Then, t_i and t_i^* can be defined as follows:

$$\begin{aligned} t_x &= (T_x - G_x)/G_w, \quad t_y = (T_y - G_y)/G_h. \\ t_w &= log(T_w/G_w), \quad t_h = log(T_h/G_h). \\ t_x^* &= (T_x^* - G_x)/G_w, \quad t_y^* = (T_y^* - G_y)/G_h. \\ t_w^* &= log(T_w^*/G_w), \quad t_h^* = log(T_h^*/G_h). \end{aligned} \tag{3}$$

Here, x, y, w and h represent the coordinates, width and height of the bounding box. T, G and T^* are the predicted box, the anchor box and the ground-truth box [11], respectively.

B. Basic Definitions

Basic definitions of the data format, data split and secure computation protocols in SecRCNN are given as follows.

Data Format: In SecRCNN, we adopt a modified fixed-point format [13] to store and transmit random shares. In this format, an arbitrary fixed-point number x is represented as $x = (-1)^s \cdot \hat{x} \cdot 10^{-c}$, where $s \in \{0, 1\}$ is the sign of x. For the original data (i.e., medical image pixels) and the random share, $\hat{x}$ belongs to two different groups with different prime orders, which are $Z_q = \{0, 1, \ldots, q-1\}$ and $Z_p = \{0, 1, \ldots, p-1\}$. Under most circumstances, the image pixels are integers ranged from 0 to 255. To guarantee the

security level, we usually set p to be a large prime, $q > 255$ and $p \gg q$. $c > 2$ is a fixed integer that controls the representation precision; c is a public parameter. According to the common statistical theorem [14], multiplying the uniformly random value $(-1)^{s} \cdot \hat{x}$ by a constant 10^{-c} does not change its original distribution, which means that the multiplication does not lead to more information about the original data revealed to the adversary. Thus, the publication of c does not influence the security of SecRCNN. Note that all random shares are represented as the modified fixed-point data format in the paper. For brevity, we do not specifically indicate the format again.

Data Split: Data split is then used to split medical images into random shares. For most cases, each pixel of an image is stored as a number of integers, e.g., grey values or RGB values. However, pixels of the processed images are sometimes stored as floating point numbers. For unity and security, the pixels are uniformly transformed into the fixed-point format while performing data split. Given an arbitrary medical image pixel $\hat{x}_0$, it is first rounded to c decimals. The rounded result is a fixed-point number and stored as $x = (-1)^{s} \cdot \hat{x} \cdot 10^{-c}$. To split the pixel, the data owner then uniformly selects a random value $\hat{x}'$ from Z_p and a random sign $s' \in \{0, 1\}$. Next, he computes $\hat{x_1} = (-1)^{s} \cdot \hat{x} - (-1)^{s'} \cdot \hat{x}'$. If $\hat{x_1} < 0$, $s'' = 1$; otherwise, $s'' = 0$. Finally, $\hat{x}''$ is obtained by calculating the absolute value $|\hat{x_1}|$ mod p. Thus, x is split into two random shares $x' = (-1)^{s'} \cdot \hat{x}' \cdot 10^{-c}$ and $x'' = (-1)^{s''} \cdot \hat{x}'' \cdot 10^{-c}$. To recover the original value, we can simply compute $x = x' + x'' = (-1)^{s'} \cdot \hat{x}' \cdot 10^{-c} + (-1)^{s''} \cdot \hat{x}'' \cdot 10^{-c}$. An example of a medical image split is given in Appendix.

Secret Sharing Protocol: All secret sharing protocols in this paper satisfy the following formal definition. Suppose that $G_{p,q}(\mathcal{X}, \mathcal{S})$ is an arbitrary secret sharing protocol. Given the random shares of inputs $\mathcal{X} = \{(x_1', x_1''), (x_2', x_2''), \ldots\}$ and two edge servers $\mathcal{S} = \{S_1, S_2\}$, $G_{p,q}$ outputs two random shares f' and f'' of the computation result. To recover the computation result f, one needs to compute $f = f' + f''$.

C. Basic Secure Protocols

The following sub-protocols, proposed in [15], are the basic components to complete the secure linear and nonlinear functions of SecRCNN. The sub-protocols are operated between two edge servers. Their implementation details are presented in Section IX-B.

Secure Comparison Protocol (SCmp): given the random shares of two inputs (μ', μ'') and (υ', υ''), it performs secure comparison and outputs (f', f''), where $f' + f'' = 0$ if $\mu > \upsilon$; otherwise, $f' + f'' = 1$. During execution, two intermediate values are exchanged between the two edge servers.

Secure Addition Protocol (SAdd): given the random shares of two inputs (μ', μ'') and (υ', υ''), it performs secure addition and outputs (f', f''), where $f' + f'' = \mu + \upsilon$. During execution, no intermediate values are exchanged.

Secure Multiplication Protocol (SMul): given the random shares of two inputs (μ', μ'') and (υ', υ''), it performs secure multiplication and outputs (f', f''), where $f' + f'' = \mu \cdot \upsilon$. During execution, four intermediate values are exchanged between the two edge servers.

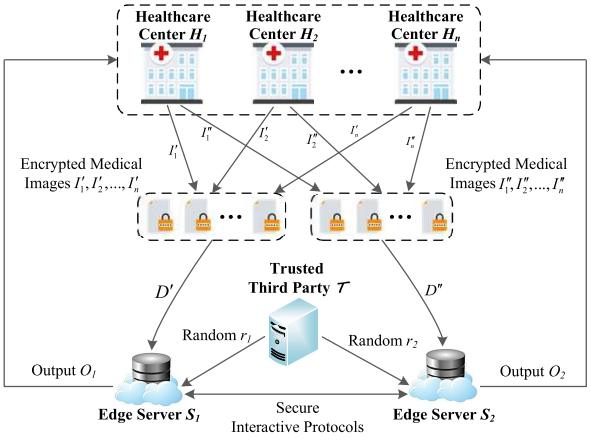


Fig. 1. System model of SecRCNN.

III. System Model & Attack Model

A. System Model

As illustrated in Fig.1, four types of participants comprise SecRCNN, namely healthcare centers $\mathcal{H} = \{H_1, H_2, \ldots, H_n\}$, two edge servers S_1 and S_2, and the trusted third party $\mathcal{T}$.

- $H_1, H_2, \ldots, H_n$ are healthcare centers, which would like to cooperate to train a Faster R-CNN based medical image object detection model. They are not willing to share their patients' pathology image database with others. Therefore, in SecRCNN, we randomly split the database $I_1, I_2, \ldots, I_n$ into $\{(I_1', I_1''), (I_2', I_2''), \ldots, (I_n', I_n'')\}$. The encrypted data $D' = \{I_k' | k \in \{1, 2, \ldots, n\}\}$ and $D'' = \{I_k'' | k \in \{1, 2, \ldots, n\}\}$ are then sent to the two edge servers for storage and computation.
- S_1 and S_2 are two outsourced edge servers' which are responsible for the intensive computation task. In SecRCNN, they complete all the computations of Faster R-CNN without knowing any plaintext of medical images. The final outputs O' and O'' are simultaneously sent to $H_1, H_2, \ldots, H_n$ using secure communication channels. The healthcare centers can obtain the original output by computing $O = O' + O''$.
- $\mathcal{T}$ is a trusted third server, which serves as a random value generator.

B. Attack Model

In SecRCNN, we adopt the *curious-but-honest* model. According to the model, $\mathcal{H} = \{H_1, H_2, \ldots, H_n\}$, S_1 and S_2 are all *curious-but-honest* parties. Literally, they follow the steps to complete the protocols, yet never refuse to know the data belonging to others that can benefit themselves.

Furthermore, we assume that there is a simulator ζ, which can generate uniformly random values and obtain the real view $\mathcal{V}_1$ of the secret sharing protocol. Based on $\mathcal{V}_1$, ζ tries to generate a simulated view $\mathcal{V}_2$ in polynomial time. For adversary $\mathcal{A}$, a successful attack means that $\mathcal{A}$ can find a probabilistic polynomial algorithm to distinguish $\mathcal{V}_1$ and $\mathcal{V}_2$.

Also, we hypothesize that the edge servers cannot collude with each other or be simultaneously corrupted. This hypothesis is essential because the plaintext data can be recovered by simply adding the corresponding two random shares together. In addition, we assume that there is a trusted third server that is responsible for generating uniformly random values. Note that the above assumptions are commonly used in the additive secret sharing based privacy-preserving schemes [16], [17].

IV. Secret Sharing Based Sub-Protocols

Diving to the bottom, the object detection with Faster R-CNN is completed by a series of mathematical operations. In SecRCNN, to avoid medical image data revealed to the cloud servers, all the operations have to be completed in a privacy-preserving way. Therefore, based on the Coordinate Rotation Digital Computer Algorithm (CORDIC) and the additive secret sharing technique, we design several secure computation sub-protocols, which are secure division protocol (SDiv), secure logarithm protocol (SLog) and secure exponentiation protocol (SExp). Compared with existing protocols [18], the newly proposed protocols can reduce the communication overhead while maintaining a low number of computation errors.

A. Secure Iteration of CORDIC

In the section, we implement the secure iteration process of CORDIC. CORDIC was first proposed by Volder *et al.* in [19]. There are three models of CORDIC iterative methods that we consider, namely secure linear vectoring mode (LiVec), secure hyperbolic vectoring mode (HypVec) and secure hyperbolic rotation mode (HypRot). These models work on different types of coordinate systems (rectangular coordinates for vectoring or polar coordinates for rotation) and make the coordinates rotate on different graphs (linear or hyperbolic). Based on different operation modes, CORDIC can be utilized to approximate several mathematical functions with only addition and shift operations.

Protocol 1 Secure Linear Vectoring Mode Iteration

Input: S_1 has input μ_1', υ_1' and ϑ_1'; S_2 has input μ_1'', υ_1'' and ϑ_1''; The maximum iteration number is m
Output: S_1 outputs ϑ_m'; S_2 outputs ϑ_m'';
1: Set a public known index $i \leftarrow 1$.
2: **while** $i \leq m$ **do**
3: $(\tau_i', \tau_i'') \leftarrow 2\mathtt{SCmp}(\upsilon_i, 0) - 1$.
4: $\upsilon_{i+1}' \leftarrow \upsilon_i' + \tau' \cdot \mu_1' \cdot 2^{-i}$ and $\upsilon_{i+1}'' \leftarrow \upsilon_i'' + \tau'' \cdot \mu_1'' \cdot 2^{-i}$.
5: $\vartheta_{i+1}' \leftarrow \vartheta_i' - \tau_i' \cdot 2^{-i}$ and $\vartheta_{i+1}'' \leftarrow \vartheta_i'' - \tau_i'' \cdot 2^{-i}$.
6: go for next iteration.
7: **end while**

Secure Linear Vectoring Mode Iteration. To complete the computation of LiVec, we have to introduce four variants: μ_i, υ_i, ϑ_i and τ_i. μ_i and υ_i represent the coordinates of the target vector after i times of vector rotation. ϑ_i denotes the sum of the phase position after i times of vector rotation. τ_i determines the rotation direction. As shown in Protocol 1, the value of μ_i remains equal to μ_1 during the iterative process. τ_i is computed by invoking SCmp. After getting the comparison result, S_1 and S_2 locally update $\upsilon_{i+1}' = \upsilon_i' + \tau' \cdot \mu_1' \cdot 2^{-i}$, $\upsilon_{i+1}'' = \upsilon_i'' + \tau'' \cdot \mu_1'' \cdot 2^{-i}$, $\vartheta_{i+1}' = \vartheta_i' - \tau_i' \cdot 2^{-i}$ and $\vartheta_{i+1}'' = \vartheta_i'' - \tau_i'' \cdot 2^{-i}$. It can be discovered that there are only two times intermediate values exchanges in LiVec caused by conducting SCmp. The mathematical expression of the whole calculation process is:

$$\begin{cases} \mu_{i+1} = \mu_1 \\ \upsilon_{i+1} = \upsilon_i + \tau_i \cdot \mu_1 \cdot 2^{-i} \\ \vartheta_{i+1} = \vartheta_i - \tau_i \cdot 2^{-i} \end{cases}, \tag{4}$$

and

$$\tau_i = 2\mathtt{SCmp}(\upsilon_i) - 1 = \begin{cases} 1, & \upsilon_i \leq 0 \\ -1, & \upsilon_i > 0 \end{cases}. \tag{5}$$

Protocol 2 Secure Hyperbolic Vectoring Mode Iteration

Input: S_1 has input μ_1', υ_1' and ϑ_1'; S_2 has input μ_1'', υ_1'' and ϑ_1''; The maximum iteration number is m
Output: S_1 outputs ϑ_m'; S_2 outputs ϑ_m'';
1: Set a public known index $i \leftarrow 1$.
2: **while** $i \leq m$ **do**
3: $(\tau_i', \tau_i'') \leftarrow 2\mathtt{SCmp}(\upsilon_i, 0) - 1$.
4: $\mu_{i+1}' \leftarrow \mu_i' + \tau' \cdot \upsilon_i' \cdot 2^{-i}$ and $\mu_{i+1}'' \leftarrow \mu_i'' + \tau'' \cdot \upsilon_i'' \cdot 2^{-i}$.
5: $\upsilon_{i+1}' \leftarrow \upsilon_i' + \tau' \cdot \mu_i' \cdot 2^{-i}$ and $\upsilon_{i+1}'' \leftarrow \upsilon_i'' + \tau'' \cdot \mu_1'' \cdot 2^{-i}$.
6: $\vartheta_{i+1}' \leftarrow \vartheta_i' - \tau_i' \cdot tanh^{-1}2^{-i}$ and $\vartheta_{i+1}'' \leftarrow \vartheta_i'' - \tau_i'' \cdot tanh^{-1}2^{-i}$.
7: **if** $i\%3$ is 1 **then**
8: do the i_{th} iteration again.
9: **else**
10: go for next iteration.
11: **end if**
12: **end while**

Secure Hyperbolic Vectoring Mode Iteration. Similar to LiVec, the computation of LiVec also needs four variants. As illustrated in Protocol 2, intermediate value exchanges occur only two times. Nevertheless, since the vector in LiVec rotates on a hyperbola, the trigonometric function $tanh$ is introduced to compute ϑ_{i+1}. The local update process of μ_{i+1} and ϑ_i becomes $\mu_{i+1}' = \mu_i' + \tau' \cdot \upsilon_i' \cdot 2^{-i}$, $\mu_{i+1}'' = \mu_i'' + \tau'' \cdot \upsilon_i'' \cdot 2^{-i}$, $\vartheta_{i+1}' = \vartheta_i' - \tau_i' \cdot tanh^{-1}2^{-i}$ and $\vartheta_{i+1}'' = \vartheta_i'' - \tau_i'' \cdot tanh^{-1}2^{-i}$. Note that if the iteration number satisfies $i = 3k+1$ and $k \in N^+$, the current iteration has to be repeated to ensure convergence. The whole computation process can be expressed as

$$\begin{cases} \mu_{i+1} = \mu_i + \tau_i \cdot \upsilon \cdot 2^{-1} \\ \upsilon_{i+1} = \upsilon_i + \tau_i \cdot \mu_i \cdot 2^{-i} \\ \vartheta_{i+1} = \vartheta_i - \tau_i \cdot tanh^{-1}2^{-i} \end{cases}, \tag{6}$$

and

$$\tau_i = 2\mathtt{SCmp}(\upsilon_i) - 1 = \begin{cases} 1, & \upsilon_i \leq 0 \\ -1, & \upsilon_i > 0 \end{cases}. \tag{7}$$

Secure Hyperbolic Rotation Mode Iteration. To calculate SExp, we propose HypRot, as shown in Protocol 3. Similar to LiVec, the iterative process of HypRot is completed

Protocol 3 Secure Hyperbolic Rotation Mode Iteration

Input: S_1 has input μ_1', υ_1' and ϑ_1'; S_2 has input μ_1'', υ_1'' and ϑ_1''; The maximum iteration number is m

Output: S_1 outputs μ_m' and υ_m'; S_2 outputs μ_m'' and υ_m'';

1: Set a public known index $i \leftarrow 1$.
2: **while** $i < m$ **do**
3: $(-\tau_i', -\tau_i'') \leftarrow -2\texttt{SCmp}(\vartheta_i, 0) + 1$.
4: $\mu_{i+1}' \leftarrow \mu_i' + \tau' \cdot \upsilon_i' \cdot 2^{-i}$ and $\mu_{i+1}'' \leftarrow \mu_i'' + \tau'' \cdot \upsilon_i'' \cdot 2^{-i}$.
5: $\upsilon_{i+1}' \leftarrow \upsilon_i' + \tau' \cdot \mu_i' \cdot 2^{-i}$ and $\upsilon_{i+1}'' \leftarrow \upsilon_i'' + \tau'' \cdot \mu_1'' \cdot 2^{-i}$.
6: $\vartheta_{i+1}' \leftarrow \vartheta_i' - \tau_i' \cdot tanh^{-1}2^{-i}$ and $\vartheta_{i+1}'' \leftarrow \vartheta_i'' - \tau_i'' \cdot tanh^{-1}2^{-i}$.
7: **if** $i\%3$ is 1 **then**
8: do the i_{th} iteration again.
9: **else**
10: go for next iteration.
11: **end if**
12: **end while**

according to Eq. 6. The difference is that HypRot updates τ_i via the following equation.

$$\tau_i = -2\texttt{SCmp}(\vartheta_i) + 1 = \begin{cases} 1, & \vartheta_i \geq 0 \\ -1, & \vartheta_i < 0 \end{cases}. \tag{8}$$

B. Secure Division Protocol

Protocol 4 Secure Division Protocol

Input: S_1 has input μ_1', υ_1'; S_2 has input μ_1'', υ_1''; The maximum iteration number m

Output: S_1 outputs f'; S_2 outputs f'';

1: $(\eta', \eta'', \epsilon) \leftarrow \texttt{SME}(\mu)$.
2: $(\alpha', \alpha'') \leftarrow \texttt{LiVec}(\eta, 1, 0, m)$.
3: $\alpha' = \alpha' \cdot 2^{\epsilon}$ and $\alpha'' = \alpha'' \cdot 2^{\epsilon}$.
4: $(f', f'') \leftarrow \texttt{SMul}(\alpha, \upsilon)$.
5: S_1 and S_2 return f' and f'', respectively.

As shown in Protocol 4, given two random share pairings (μ', μ'') and (υ', υ''), SDiv outputs (f', f''), where $\mu = \mu' + \mu''$ and $\upsilon = \upsilon' + \upsilon''$. f' and f'' are two random shares of the division result, i.e., $f' + f'' = \upsilon/\mu$. In the process, S_1 and S_2 have to compute $\texttt{LiVec}(\mu, 1, 0)$. However, to ensure convergence, μ must be greater than 1 and less than 2. For this, we invoke the single-precision representation method SME in [18] to meet the requirement. By calculating $\eta \cdot 2^{\epsilon} \leftarrow \texttt{SME}(\mu)$, the input of LiVec is ensured to satisfy $2 > \eta > 1$. ϵ is the exponent of μ. Moreover, suppose the input size is n. Compared with the existing protocol [18], the exchanged number of messages decreases from $8n$ to $2n$.

Secure Division Protocol. In SDiv, S_1 and S_2 first convert μ into the single-precision format by invoking $(\eta', \eta'', \epsilon) \leftarrow \texttt{SME}(\mu)$, which ensures the inputs of LiVec within the valid range. Then, the reciprocal of $\eta = \eta' + \eta''$ is computed by utilizing $(\alpha', \alpha'') \leftarrow \texttt{LiVec}(\eta, 1, 0)$. Finally, the two edge servers update $\alpha' = \alpha' \cdot 2^{\epsilon}$ and $\alpha'' = \alpha'' \cdot 2^{\epsilon}$ and continue to calculate $(f', f'') \leftarrow \texttt{SMul}(\alpha, \upsilon)$. An example of SDiv is presented in Appendix for a better understanding of the workflow of our secret sharing protocols. For brevity, the examples of the other two protocols are omitted.

C. Secure Natural Logarithm Protocol

Protocol 5 Secure Natural Logarithm Protocol

Input: S_1 has input μ_1'; S_2 has input μ_1''; The maximum iteration number m

Output: S_1 outputs f'; S_2 outputs f'';

1: $(\eta', \eta'', \epsilon) \leftarrow \texttt{SME}(\mu)$.
2: $(\alpha', \alpha'') \leftarrow \texttt{HypVec}(\eta + 1, \eta - 1, 0, m)$.
3: $f' \leftarrow 2\alpha' + \epsilon \cdot log2$ and $f'' \leftarrow 2\alpha''$.
4: S_1 and S_2 return f' and f'', respectively.

Given the random share pairing (μ', μ''), Protocol 5 outputs (f', f''), where $\mu = \mu' + \mu''$. f' and f'' are two random shares of the logarithm result, i.e., $f' + f'' = log(\mu)$. Here, S_1 and S_2 compute $\texttt{HypVec}(\mu + 1, \mu - 1, 0)$ to complete the iterative approximation. To guarantee convergence, the condition for loop termination in SME becomes that the mantissa η has to be between 0.1069 and 9.3573. Compared with the existing protocol [18], the number of messages required to be exchanged is reduced from $12n$ to $2n$.

Secure Natural Logarithm Protocol. First, the two edge servers compute $(\eta', \eta'', \epsilon) \leftarrow \texttt{SME}(\mu)$ to restrict the input of LiVec into the valid range. Then, S_1 and S_2 collaboratively invoke the secure CORDIC based method to calculate $(\alpha', \alpha'') \leftarrow \texttt{HypVec}(\eta + 1, \eta - 1, 0)$. Finally, they update $f' \leftarrow 2\alpha' + \epsilon \cdot log2$ and $f'' \leftarrow 2\alpha''$.

D. Improved Secure Natural Exponentiation Protocol

Protocol 6 Secure Natural Exponentiation Protocol

Input: S_1 has input μ_1'; S_2 has input μ_1''; The maximum iteration number m

Output: S_1 outputs f'; S_2 outputs f'';

1: $\mu' = \alpha' + \beta'$ and $\mu'' = \alpha'' + \beta''$, where α' and α'' are integers and $0 < \beta' < 1$, $0 < \beta'' < 1$.
2: $[(\gamma', \gamma''), (\delta', \delta'')] \leftarrow \texttt{HypRot}(1/R_n, 0, \beta, m)$, where $R_n = \prod_{i=1}^{n-1} \sqrt{1 - 2^{-2i}}$.
3: S_1 computes $a \leftarrow e^{\alpha'}$ and splits it into random shares $a \leftarrow a' + a''$.
4: S_2 computes $b \leftarrow e^{\alpha''}$ and splits it into random shares $b \leftarrow b' + b''$.
5: S_1 sends a'' to S_2 and S_2 sends b' to S_1.
6: $(\varrho', \varrho'') \leftarrow \texttt{SMul}(a, b)$.
7: $(f', f'') \leftarrow \texttt{SMul}(\varrho, \gamma + \delta)$.
8: S_1 and S_2 return f' and f'', respectively.

As shown in Protocol 6, SExp implements the iterative approximation process of natural exponentiation based on HypRot. Similar to LiVec, there is also an input range restriction for HypRot which is $[-1.1181, 1.1181]$. f' and f'' are two random shares of the exponentiation result, i.e.,

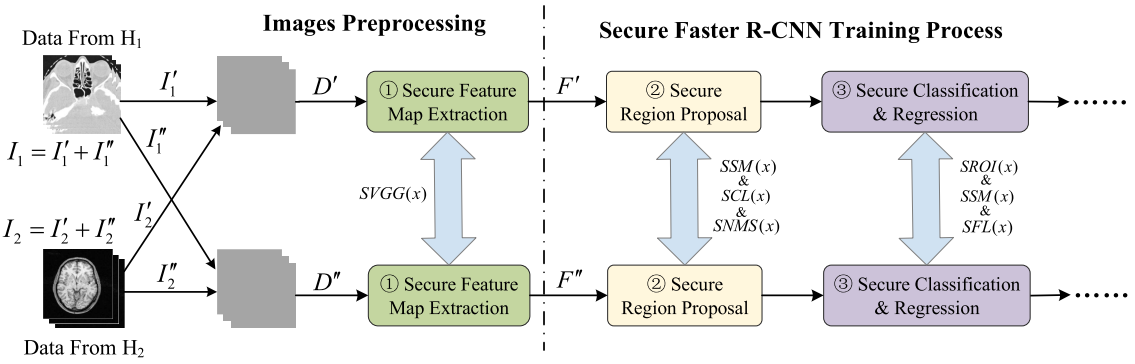


Fig. 2. Privacy-preserving faster R-CNN for object detection of medical images.

$f' + f'' = e^{\mu}$. Compared with the existing protocol [18], the exchanged number of messages is reduced from $4n$ to $2n$.

Secure Natural Exponentiation Protocol. The two edge servers first split the inputs into $\mu' = \alpha' + \beta'$ and $\mu'' = \alpha'' + \beta''$, where α' and α'' are integers, and β' and β'' are the decimal parts of μ. By computing $(\gamma', \gamma'', \delta', \delta'') \leftarrow \texttt{HypRot}(1/R_n, 0, \beta, m)$, we have $\gamma' + \gamma'' + \delta' + \delta'' = e^{\beta}$. Here, $1/R_n = \prod_{i=1}^{n-1} \sqrt{1 - 2^{-2i}} \approx 0.8281$, when $n \to \infty$. S_1 and S_2 then compute $a = e^{\alpha'}$, $b = e^{\alpha''}$, and split them into random shares (a', a'') and (b', b''). Subsequently, a'' and b' are exchanged between S_1 and S_2. Finally, SMul is invoked twice for computing $a \cdot b$ and $\varrho \cdot (\gamma + \delta)$, where $\varrho = \varrho' + \varrho''$ is the multiplication result of a and b. The outputs of the second invocation for SMul, f' and f'', are two random shares of the natural exponentiation result of μ.

V. Privacy-Preserving Object Detection of Medical Images

In this section, we provide the implementation details of SecRCNN and discuss the feasibility of extending SecRCNN to a multiparty setting.

A. High-Level Overview of SecRCNN

Before introducing the implementation details of SecRCNN, we first give its high-level overview for a better understanding of its workflow, shown in Fig.2. SecRCNN is composed of three stages, namely secure feature map extraction, secure region proposal and secure regression and classification. The overall goal of SecRCNN is to implement object detection of medical images without revealing any information to the original images revealed to the edge servers. To achieve this goal, SecRCNN first splits medical image pixels into random shares and uploads the random shares to the edge servers. Then, all subsequent computations are performed on secretly shared pixels using our secure protocols. An overview of each stage is below.

Secure Feature Map Extraction. To ensure no plaintext image pixel information is revealed to the edge servers, the feature map extraction of medical images is completed via the secure feature extraction network (SVGG). The input of SVGG is the secretly shared pixel maps (RGB or grey values) of medical images with a fixed size, i.e., D' and D'', which are uploaded by the healthcare centers. SVGG can be based on an arbitrary ImageNet [20], but the involved three basic neural layers have to be implemented with the three secure protocols, which are secure convolutional layer (SCL), secure ReLU layer (SRL) and secure pooling layer (SPL). In this paper, we choose VGG as the feature extractor. After completing SVGG, S_1 and S_2 obtain the shared feature maps $\mathcal{F}''$ and $\mathcal{F}''$, respectively.

Secure Region Proposal. Two parts comprise the secure region proposal stage, namely the anchor generation layer (AGL) and the secure PRN network (SPRN). The goal of the region proposal stage is to operate anchor recommendation without leaking any medical image feature information. To achieve this goal, SecRCNN first calculates the AGL with a specially designed intersection over union (IoU) algorithm and gets a series of candidate anchors. Then, it recommends good anchors (i.e., proposal regions) by invoking two SCLs, one secure softmax function (SSM) and one secure non-maximum suppression protocol (SNMS). Finally, the recommended proposal regions are pushed to the next stage.

Secure Regression & Classification. In the last stage, the destination of SecRCNN is to complete the bounding box regression and object classification task without disclosing the image feature information of the recommended proposal regions. Both the regression and classification processes are accomplished by a modified SPRN, in which the two SCLs are substituted with two SFLs. During the process, the input proposal regions are reshaped to a fixed size by the secure regions of interest protocol (SROI).

B. Secure Feature Map Extraction

Before the medical images are sent to SRPN, they are applied to SVGG to extract the feature map. SVGG consists of three kinds of layers: SCL, SRL and SPL. Given the input matrix $x = (x_{0,0}, x_{0,1}, \ldots, x_{w,h})$, SCL lets S_1 and S_2 compute $\phi' = \sum_{i=0}^{w-1} \sum_{j=0}^{h-1} \omega'_{i,j} x'_{i,j} + b$ and $\phi'' = \sum_{i=0}^{w-1} \sum_{j=0}^{h-1} \omega''_{i,j} x''_{i,j}$, where $\omega_{i,j}$ and b are publicly known weight parameter and bias. The addition here is performed locally by invoking SAdd. (ϕ', ϕ'') is the output of SCL.

In SRL, we first utilize $s = (-\texttt{SCmp}(x_{i,j}, 0) + 2)/2$ to determine the sign of the input. If $s = 0$, SRL outputs $(x'_{i,j}, x''_{i,j})$; otherwise, it outputs $(\alpha x'_{i,j}, \alpha x''_{i,j})$. α is a common learnable

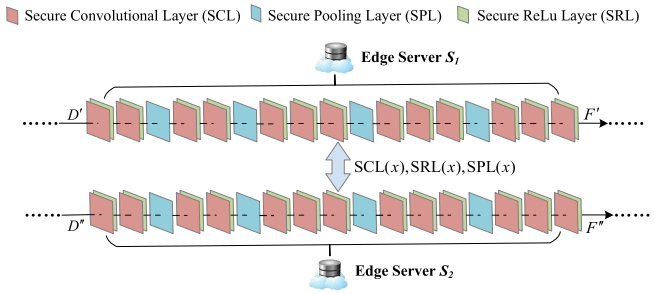


Fig. 3. Privacy-preserving VGG feature extraction architecture.

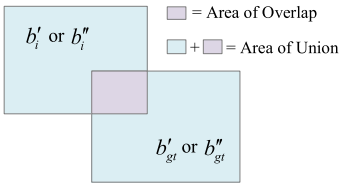


Fig. 4. The calculation for Intersection over Union in S_1 and S_2.

parameter. Specially, for convenience of parallel computing, extra computations are operated on the output of SCmp to make s to be only 0 or 1. As for SPL, each convolutional sliding window ξ_k outputs $(x_i', x_i'') = \arg\max \xi_k(x)$ by letting the two edge servers compute $(-\mathtt{SCmp}(x_i - x_j, 0) + 2)/2$ for several times. As shown in Fig.3, 13 SCLs, 13 SRLs and 4 SPLs are deployed to implement SVGG. Interested readers can refer to [15] for the details of the three secure neural layers.

C. Secure Region Proposal

Given the medical image feature map $\mathcal{F}$ output by SVGG, SRPN produces a set of candidate regions called anchors. The anchors are ranked and filtered in SRPN for use in the next stage.

1) Anchor Generation Layer: The AGL scans the medical image feature map and outputs nine kinds of anchor boxes with varying sizes $\{8, 16, 32\}$ and aspect ratios $\{0.5, 1, 2\}$. Let $\mathcal{F}'$ and $\mathcal{F}''$ be two secretly shared feature maps possessed by S_1 and S_2. We set the default stride length to 16 and the size of $\mathcal{F}$ to $w \times h$ and place it at every grid location of $\mathcal{F}'$ and $\mathcal{F}''$. The two edge servers can separately obtain the sets of anchor boxes $\mathcal{B}' = \{b_1', b_2', \ldots, b_n'\}$ and $\mathcal{B}'' = \{b_1'', b_2'', \ldots, b_n''\}$, where $n = \frac{w}{16} \times \frac{h}{16} \times 9$ and the original anchor box is $b_i = b_i' + b_i''$.

To filter and label the generated anchor boxes, we have to calculate the IoU of anchors. As shown in Fig.4, the computation process of IoU does not need data exchange and can be locally completed by letting S_1 and S_2 compute

$$IoU(b_i, b_{gt}) = \frac{Area_{overlap}(b_i', b_{gt}')}{Area_{union}(b_i', b_{gt}')}, \tag{9}$$

and

$$IoU(b_i, b_{gt}) = \frac{Area_{overlap}(b_i'', b_{gt}'')}{Area_{union}(b_i'', b_{gt}'')}. \tag{10}$$

b_{gt} is the ground-truth bounding box feature map. Given the positive IoU threshold Θ_{pos} and negative IoU threshold

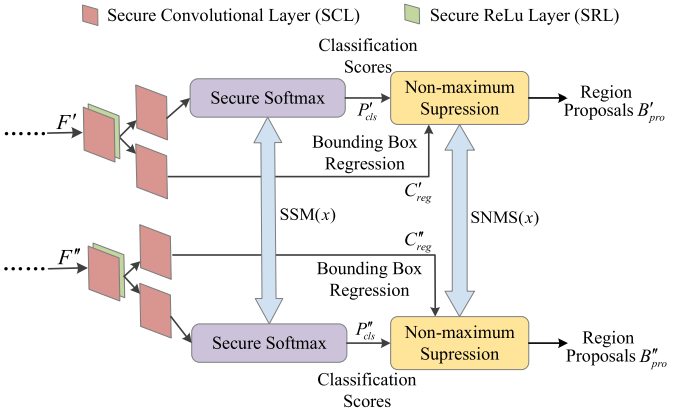


Fig. 5. Privacy-preserving region proposal layer.

Θ_{neg}, the anchor boxes that are beyond the boundary or satisfy $\Theta_{neg} < IoU_{b_i} < \Theta_{pos}$ are ignored in the following computation. The boxes satisfying $IoU(b_i, b_{gt}) > \Theta_{pos}$ are masked as "positive". The others are masked as "negative".

2) Region Proposal Network: In this step, SRPN filters $\mathcal{B}$ and recommends the good regions for the ROI layer. To train SRPN and rank the region proposals, two SCLs are invoked. One is used to compute the probability O_{cls} of each anchor being foreground or background. The other is used to compute the regression coefficients C_{reg}. The two layers are simultaneously operated and have the same kernel size (1×1). However, the one for classification has $n \times 2$ output channels, and the other has $n \times 4$ output channels. Then, O_{cls} is further processed by SSM to get the final scores P_{cls}. SSM can be expressed as

$$p_i' = SSM(i) = \mathtt{SDiv}(\mathtt{SExp}(O_{cls}(i)'), \mathtt{SExp}(O_{cls}'(0)) + \mathtt{SExp}(O_{cls}'(1))), \tag{11}$$

and

$$p_i'' = SSM(i) = \mathtt{SDiv}(\mathtt{SExp}(O_{cls}(i)''), \mathtt{SExp}(O_{cls}''(0)) + \mathtt{SExp}(O_{cls}''(1))), \tag{12}$$

where $p_i = p_i' + p_i'' = P_{cls}(i)$ is a probability vector.

Furthermore, to reduce the number of candidate anchor boxes and improve efficiency, the SNMS is deployed to select the anchor boxes whose scores are lower than others. As illustrated in Protocol 7, SNMS can be simply implemented by the SCmp based sort function and the secure IoU function.

3) Loss Function: As mentioned before, the loss function of RPN is composed of two parts, the log loss part for classification and the smooth L_1 part for box bounding regression. To calculate log loss, S_1 and S_2 have to compute

$$L_{Log}(p_i', y_i') = \mathtt{SMul}(y_i', \mathtt{SLog}(p_i')) + \mathtt{SMul}(1 - y_i', \mathtt{SLog}(1 - p_i')), \tag{13}$$

and

$$L_{Log}(p_i'', y_i'') = \mathtt{SMul}(y_i'', \mathtt{SLog}(p_i'')) + \mathtt{SMul}(y_i'', \mathtt{SLog}(p_i'')), \tag{14}$$

Protocol 7 Secure Non-Maximum Suppression Protocol

Input: S_1 has the set of candidate regions Γ_0' and corresponding scores Ω_0'; S_2 has the set of candidate regions Γ_0'' and corresponding scores Ω_0''; common parameter IoU threshold is Θ;

Output: S_1 outputs the filtered regions Γ'; S_2 outputs the filtered regions Γ'';

if $\Gamma_0 = \Gamma_0' + \Gamma_0'' \notin \mathcal{B}$ or $\Omega_0 = \Omega_0' + \Omega_0'' \notin S_{cls}$ **then**
  Return error.
end if
Update $(\Omega_0', \Omega_0'', \Gamma_0', \Gamma_0'') \leftarrow sort(\Omega_0, \Gamma_0)$.
Set the retained candidate regions $(\Gamma', \Gamma'') = \varnothing$.
while Ω_0 is not empty **do**
  $(\omega_0', \omega_0'') \leftarrow \Omega(0)$ and $(\tau_0', \tau_0'') \leftarrow \Gamma(0)$.
  Update $(\Gamma', \Gamma'') \leftarrow \Gamma + \tau_0$.
  Delete τ_0 from Γ and ω_0 from Ω.
  for all $\tau_i \in \Gamma$ **do**
    cmp =
    if $IoU(\tau_i, \tau_0) > \Theta$ **then**
      Delete τ_i from Γ and ω_i from Ω.
    end if
  end for
end while
S_1 and S_2 return Γ' and Γ'', respectively.

where $y_i = y_i' + y_i''$ is the ground-truth label and p_i is the prediction probability. For smooth L_1 loss, let the two edge servers compute

$$Smooth_{L_1}(x_i') = \begin{cases} \frac{1}{2}\sigma^2 \cdot \texttt{SMul}(x_i', x_i'), & |x| < \frac{1}{\sigma^2} \\ sign(x_i') \cdot x_i' - \frac{1}{2\sigma^2}, & otherwise, \end{cases} \quad (15)$$

and

$$Smooth_{L_1}(x_i'') = \begin{cases} \frac{1}{2}\sigma^2 \cdot \texttt{SMul}(x_i'', x_i''), & |x| < \frac{1}{\sigma^2} \\ sign(x_i'') \cdot x_i' - \frac{1}{2\sigma^2}, & otherwise, \end{cases} \quad (16)$$

where

$$sign(x) = -2\texttt{SCmp}(x, 0) + 1. \quad (17)$$

Here, $x_i = x_i' + x_i''$ is the difference between the predicted and ground-truth regression coordinates $c_i - t_i$. σ is a predefined constant. Then, the objective loss L can be computed as

$$L' = \frac{1}{N_c}\sum_{i=0}^{N_c} L_{Log}(p_i', y_i') + \frac{\lambda}{N_r}\sum_{i=0}^{N_r} \texttt{SMul}(y_i', Smooth_{L_1}(c_i' - t_i')), \quad (18)$$

and

$$L'' = \frac{1}{N_c}\sum_{i=0}^{N_c} L_{Log}(p_i'', y_i'') + \frac{\lambda}{N_r}\sum_{i=0}^{N_r} \texttt{SMul}(y_i'', Smooth_{L_1}(c_i'' - t_i'')), \quad (19)$$

where $c_i \in C_{reg}$ and λ is a constant that is set to 3 by default. In Section II, the computation method and meaning of the four types of t_i are listed. They can be locally computed by S_1 and S_2.

D. Secure Regression & Classification

Protocol 8 Secure Regions of Interest Protocol

Input: S_1 has the feature maps of region proposals $\mathcal{X}'$; S_2 has the feature maps of region proposals $\mathcal{X}''$; The fixed width w and height h are public parameters.

Output: S_1 outputs the reshaped region proposals $\mathcal{Y}'$; S_2 outputs the reshaped region proposals $\mathcal{Y}''$;

S_1 and S_2 average divide $\mathcal{X}'$ and $\mathcal{X}''$ into $w \times h$ parts, $(\mathcal{X}_1', \ldots, \mathcal{X}_{w\times h}')$ and $(\mathcal{X}_1'', \ldots, \mathcal{X}_{w\times h}'')$.
for $1 < i < w \times h$ **do**
  S_1 and S_2 compute $(x', x'') \leftarrow \arg\max_{x_j \in \mathcal{X}_i} \mathcal{X}_i(j)$.
  Add x' and x'' into $\mathcal{Y}'$ and $\mathcal{Y}''$.
end for
S_1 and S_2 return $\mathcal{Y}'$ and $\mathcal{Y}''$, respectively.

In this stage, SecRCNN invokes `SROI` to reshape the size of the region proposals produced by SRPN, and two fully-connected layers to generate the final output. As mentioned before, there can be nine types of proposal anchor boxes. The different sizes make it difficult to directly deploy them as the input for the following neural layer computation. Therefore, as shown in Protocol 8, the region proposals are further reshaped into a fixed size $w \times h$. Then, SecRCNN computes the classification and regression coefficients in a similar way to SRPN. The differences are that the regions are further simplified by SRPN and SCL is changed to SFL. The computation method for SFL is the same as for SCL.

E. Feasibility for Multiparty Computation

For a better understanding of the SecRCNN workflow, only the two-party setting (i.e., two edge servers) is discussed above. However, two parties are sometimes not enough in applications that need to use SecRCNN. Thus, in this section we discuss the feasibility of extending SecRCNN to a multi-party setting (MPC).

To expand to MPC, we have to adapt the secure protocols of SecRCNN to allow for multiparty computation. According to the definitions of the interactive protocols, it can be discovered that all of them are composed of three kinds of basic sub-protocols: `SAdd`, `SCmp` and `SMul`. Therefore, proving the feasibility of SecRCNN for MPC is equivalent to proving that the three protocols support MPC. Among the three protocols, `SAdd` can be locally completed. Therefore, it is trivial to state the feasibility of `SAdd` to extend to MPC. `SCmp` is based on the most significant bit (MSB) protocol, which has been proved to support MPC [21]. Similar to `SCmp`, the basis of `SMul`, Beaver's triplet, has also been originally designed for supporting MPC [22]. In conclusion, if required, it is completely feasible to make SecRCNN to operate in a multiparty setting.

TABLE I
COMPUTATIONAL COMPLEXITY OF EACH PROTOCOL IN SECRCNN

Protocol	Our Protocols	[18]
LiVec, HypVec, HypRot	$\mathcal{O}(nm)(T_{\texttt{SCmp}} + T_{\texttt{Mul}})$	N.A.
SDiv	$\mathcal{O}(nm)(T_{\texttt{SCmp}} + T_{\texttt{Mul}}) + \mathcal{O}(n)T_{\texttt{Mul}} + \mathcal{O}(1)T_{\texttt{SMul}}$	$\mathcal{O}(nm)(T_{\texttt{SCmp}} + T_{\texttt{SMul}}) + \mathcal{O}(n)T_{\texttt{Mul}} + \mathcal{O}(1)T_{\texttt{SMul}}$
SLog	$\mathcal{O}(nm)(T_{\texttt{SCmp}} + T_{\texttt{Mul}}) + \mathcal{O}(n)T_{\texttt{Mul}}$	$\mathcal{O}(nm)(T_{\texttt{SCmp}} + T_{\texttt{SMul}}) + \mathcal{O}(n)T_{\texttt{Mul}}$
SExp	$\mathcal{O}(nm)(T_{\texttt{SCmp}} + T_{\texttt{Mul}}) + \mathcal{O}(1)T_{\texttt{SMul}}$	$\mathcal{O}(nm)(T_{\texttt{SCmp}} + T_{\texttt{SMul}}) + \mathcal{O}(1)T_{\texttt{SMul}}$
SNMS	$\mathcal{O}(n \log n + n^2)T_{\texttt{SCmp}}$	N.A.
SROI	$\mathcal{O}(n)T_{\texttt{SCmp}}$	N.A.

VI. THEORETICAL ANALYSIS OF SECRCNN

A. Computational Complexity

To evaluate the efficiency of SecRCNN, we analyse the computational complexity of each sub-protocol, as shown in Table I. The input size of each sub-protocol is n; m is the maximum iteration number of LiVec, HypVec and HypRot; $T_{\texttt{Mul}}$, $T_{\texttt{SMul}}$ and $T_{\texttt{SCmp}}$ are the runtimes of local multiplication, secure multiplication and secure comparison functions, where $T_{\texttt{Mul}} < T_{\texttt{SMul}} < T_{\texttt{SCmp}}$. For the division, logarithm and exponentiation sub-protocols, we compare their computation complexities with the existing ones [18].

For LiVec, HypVec and HypRot, all three protocols perform one SCmp and four or six local multiplications at each iteration, which takes $\mathcal{O}(nm)(T_{\texttt{SCmp}} + T_{\texttt{Mul}})$ time. For SDiv, SLog and SExp, what they have in common is that all of them contain a CORDIC based iterative protocol during their operation processes (LiVec, HypVec or HypRot). The difference is that SExp does not operate SME which takes $\mathcal{O}(n)T_{\texttt{Mul}}$ time. Additionally, SLog does not have to do SMul after completing the iterative protocol. Compared with the existing protocols of the three mathematical functions, our protocols substitute SMul with local multiplication, which can reduce the runtime for iteration. For SNMS, its computational complexity is $\mathcal{O}(n \log n + n^2)T_{\texttt{SCmp}}$. $\mathcal{O}(n \log n)T_{\texttt{SCmp}}$ time is spent on quicksort. $\mathcal{O}(n^2)T_{\texttt{SCmp}}$ time is spent on the candidate region filtering. For SROI, $\mathcal{O}(n)T_{\texttt{SCmp}}$ time is spent on scanning the whole input space and selecting the maximum feature value through SCmp.

The three stages of SecRCNN are completed by invoking the above protocols, or the previously proposed two sub-protocols. Thus, the computational complexity of the three stages can be easily derived from the linear combination of the protocols. For brevity, we omit their analysis.

B. Correctness Analysis of SecRCNN

In SecRCNN, all of the medical images in D are split into two random shares D' and D''. Intuitively, we cannot ensure that the outputs of SecRCNN satisfy $f_{cls} = f'_{cls} + f''_{cls}$ and $f_{reg} = f'_{reg} + f''_{reg}$, where f_{cls} and f_{reg} are the original outputs of Faster R-CNN. Therefore, the following proof is given to state the correctness of SecRCNN.

Theorem 1: The computation of feature map extraction, proposal region network and classification & regression in SecRCNN is correct.

Proof: First, the feature map extraction is composed of SCL, SRL and SPL. Among the three types of neural layers, SRL and SPL are implemented by calling the SCmp multiple times. Since the SCmp never changes the input value and only outputs the comparison result 1 or -1, SRL and SPL are both errorless. For SCL, SMul is used to complete the convolutional operations. SMul is a previously proposed protocol and has been proved to be correct. Therefore, SCL is also errorless. Overall, it can be deduced that the feature map extraction of SecRCNN is correct. Second, besides SMul, SCmp and the local computation that cannot influence the correctness, SDiv, SExp and SLog are deployed to compute the softmax function and loss function in SRPN. The CORDIC iteration methods used in SDiv, SExp and SLog can maintain the convergence speed at one bit per iteration. We set the default iteration number to be a small value, like 10. The computation error can reach a degree of no more than 10^{-10} which is completely negligible for application purposes. According to the preliminary assumption, the three protocols are considered to be correct here. Consequently, SRPN can be proved to be correct. Finally, for the classification and regression process, the involved sub-protocols are identical to SRPN. Naturally, the process is also correct. In conclusion, the correctness of SecRCNN is proved. □

C. Security Analysis of SecRCNN

The security analysis of SecRCNN is based on the following definitions and lemmas.

Definition 1: We say that a protocol π is secure if there exists a probabilistic polynomial-time simulator ζ that can generate a view for the adversary $\mathcal{A}$ in the real world and the view is computationally indistinguishable from its real view.

Lemma 1 [23]: A protocol is perfectly simulatable if all its sub-protocols are perfectly simulatable.

Lemma 2 [14]: If a random element r is uniformly distributed on Z_n and independent from any variable $x \in Z_n$, then $r \pm x$ is also uniformly random and independent from x.

According to *Definition 1* and *Lemma 1*, to prove the security of SecRCNN, we have to prove that all simulated views of its sub-protocols are computationally distinguishable from their real views. The proofs are given as follows.

Theorem 2: The protocols LiVec, HypVec and HypRot are secure in the semi-honest model.

Proof: S_1 has the same type of real view for LiVec, HypVec and HypRot, $view_1 = \{m, \mathcal{C}, \mathcal{D}, \mathcal{E}, \mathcal{J}\}$, where

$\mathcal{C} = \{\mu'_1, \mu'_2, \ldots, \mu'_m\}$, $\mathcal{D} = \{\upsilon'_1, \upsilon'_2, \ldots, \upsilon'_m\}$, $\mathcal{E} = \{\theta'_1, \theta'_2, \ldots, \theta'_m\}$, $\mathcal{J} = \{\tau'_1, \tau'_2, \ldots, \tau'_m\}$. μ'_1, υ'_1 and θ'_1 are uniformly random inputs selected in Z_p,. $\tau'_i \in \mathcal{J}$ are outputs of SCmp. m is a public available constant. Since SCmp is a previously proposed protocol proved to be secure in semi-honest model, all elements $\tau'_i \in \mathcal{J}$ are uniformly random shares as long as the elements of $\mathcal{C}_{it} = \mathcal{C} - \{\mu'_1\}$, $\mathcal{D}_{it} = \mathcal{D} - \{\upsilon'_1\}$, $\mathcal{E}_{it} = \mathcal{E} - \{\theta'_1\}$ are uniformly random. From the iterative equations mentioned in Section IV, μ'_{i+1}, υ'_{i+1}, θ'_{i+1}, $i > 0$ are obtained by locally operating first order polynomial about μ'_i, υ'_i, θ'_i. Consequently, based on the inductive method, it is easy to deduce that $\mathcal{C}_{it}$, $\mathcal{D}_{it}$ and $\mathcal{E}_{it}$ are completely composed of uniformly random values. Furthermore, we can conclude that $\mathcal{C}$, $\mathcal{D}$, $\mathcal{E}$, $\mathcal{J}$ are all uniformly random. Since the output of the three protocols for S_1 are (μ'_m, υ'_m) or θ'_m, we can also guarantee that $output_1 = \{\mu'_m, \upsilon'_m\}$ or $output_1 = \{\theta'_m\}$ is uniformly random. Therefore, both $view_1$ and $output_1$ are simulatable by simulator ζ. And for $\mathcal{A}$, it is unable to distinguish the real view and simulated view in polynomial time. In the same way, the real view of S_2 can be proved to be computationally indistinguishable from its simulated view. □

Theorem 3: The protocols SDiv, SLog and SExp are secure in the semi-honest model.

Proof: In SDiv, the real view of S_1 is $view_1 = \{\mu', \upsilon', \eta', \alpha'\}$. μ' and υ' are uniformly random inputs selected in Z_p; η' is the result of μ' multiplied with constants. Therefore, it is trivial to prove that η' is uniformly random. α' is the output of LiVec. According to *Theorem 2*, α' is also a uniformly random share in Z_p. The output view of S_1 is $output_1 = f'$, $f' =$SMul$(\upsilon', 2^{\epsilon'} \cdot$ LiVec$(\eta', 1, 0))$. Since SMul is a previously proposed secure protocol and its inputs are uniformly random values from the above analysis, f' is a uniformly random share value. As a consequence, $view_1$ and $output_1$ are simulatable for ζ and computationally distinguishable from the simulated views for $\mathcal{A}$. Likewise, the real views of S_2 are also computationally indistinguishable from its simulated view.

In SLog, the real view of S_1 is $view_1 = \{\mu', \eta', \alpha'\}$. Similar to SDiv, μ' is a uniformly random input, η' is the product of μ' times constants and α' is generated by the HypVec proved to be secure in the proof of *Theorem 2*. Naturally, the three elements are uniformly random values and simulatable. The output view of S_1 is $output_1 = f'$, $f' =$HypVec$(\eta' + 1, \eta' - 1, 0) + \epsilon \cdot \log 2$. f' is the addition of a uniformly random share and a constant. Thus, f' is also uniformly random and simulatable. And it can be deduced that ζ can generate simulated views in polynomial time which are computationally indistinguishable from the real views for $\mathcal{A}$. In the same way, it can be proved that the real view S_2 is computationally indistinguishable from its simulated view.

In SExp, the real view of S_1 is $view_1 = \{\mu', \alpha', \beta', \gamma', \delta', R_n, a, a', b', \varrho'\}$, where $\mu' = \alpha' + \beta'$. μ' is a uniformly random input. From the definition of SExp, Based on *Lemma 2*, the split shares α', β', a' and b' are all uniformly random. Both γ' and δ' are outputs of HypRot, which are uniformly random according to the proof of *Theorem 2*. ϱ' is the output of SMul, a previously proposed secure protocol. Therefore, it is also uniformly random. Moreover, R_n is a constant that cannot influence the security of SExp. The output view of S_1 is $output_1 = f'$, $f' =$SMul$($HypRot$(\frac{1}{R_n}, 0, \beta), \varrho)$. Similar to SDiv, f' is the output of SMul and a uniformly random value. As a consequence, $view_1$ and $output_1$ are simulatable and their simulated views are computationally indistinguishable from them for $\mathcal{A}$. Similarly, the view of S_2 is also simulatable and computationally indistinguishable. □

The feature map extraction, SRPN and classification & regression of SecRCNN are then proved to be secure.

Theorem 4: The interactive protocol of feature map extraction, SRPN and classification & regression process in SecRCNN are secure in the semi-honest model.

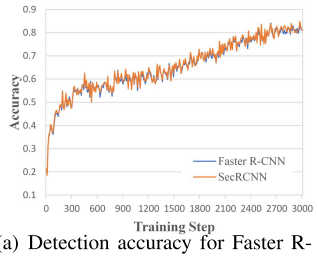
Proof: $\mathcal{A}$ eavesdrops on the transmission channels between the two edge servers and records the messages about the three interactive protocols inputs into an input tape $tape_{in}$ and outputs into an output tape $tape_{out}$. According to the definitions of the interactive protocols, $\mathcal{A}$ have $tape_{in} = view_{SCmp} \cup view_{SAdd} \cup view_{SMul} \cup view_{SDiv} \cup view_{SLog} \cup view_{SLog}$ and $tape_{out} = output_{SCmp} \cup output_{SAdd} \cup output_{SMul} \cup output_{SDiv} \cup output_{SLog} \cup output_{SLog}$. Here, the elements belonging to the same sub-protocol are pushed into the same view. Let's hypothesize that there is a polynomial-time algorithm that allows $\mathcal{A}$ to tell whether $tape_{in}$ and $tape_{out}$ are simulated by ζ. Then, it must be unable for ζ to find a polynomial-time algorithm to simulate the views $tape_{in}$ and $tape_{out}$. Based on the previously proved theorems, it is guaranteed that the views of all the sub-protocols are simulatable. Based on *Lemma 1*, $tape_{in}$ and $tape_{out}$ are simulatable. Furthermore, ζ is capable of generating views that are computational distinguishable from the real views, which is opposite to the hypothesis. Thus, it can be easily deduced that the hypothesis does not stand and the interactive protocols of SecRCNN are secure in the *curious-but-honest* model.

VII. Experiments

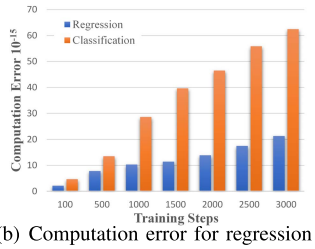
In this section, we first evaluate the performance and security of SecRCNN. Then, to further assess the effectiveness and efficiency of SecRCNN, we conduct experiments to evaluate the computation error, runtime and communication overhead of the newly proposed sub-protocols. All the experiments were conducted with the ImageCLEF medical image dataset which is publicly available and provided by the IRMA group from the University Hospital of Aachen, Germany [24]. The original data are encrypted and sent to the two edge servers on a laptop with an Intel(R) Core(TM) i5-7200 CPU @2.50GHz and 8.00GB of RAM. Two computers equipped with NVIDIA GeForce GTX 1080 TI graphic card and 8.00GB of RAM are deployed as the edge servers.

A. Performance Evaluation of SecRCNN

To evaluate the performance of SecRCNN, we randomly selected 2000 medical images for training and 500 images for validation from ImageCLEF. The batch size was set to 128. The model used to extract feature maps was the VGG-16 [12] and implemented according to [25]. p and q, mentioned in Section II-B, were set to two Mersenne primes, $p = 2^{31} - 1$

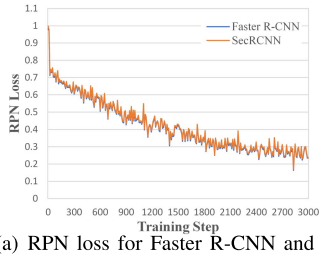


(a) Detection accuracy for Faster R-CNN and SecRCNN (3000 iterations)

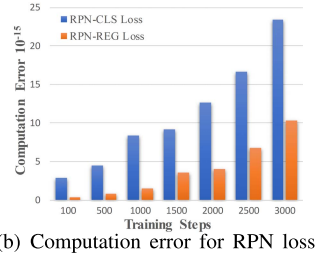


(b) Computation error for regression and classification accuracy between Faster R-CNN and SecRCNN (3000 iterations)

Fig. 6. Performance analysis of SecRCNN with accuracy.

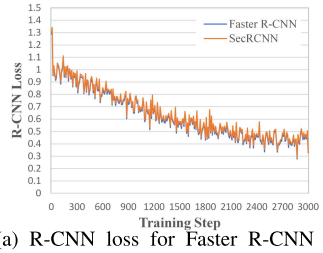


(a) RPN loss for Faster R-CNN and SecRCNN (3000 iterations)

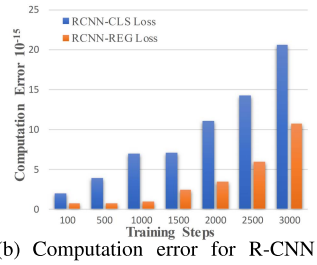


(b) Computation error for RPN loss between Faster R-CNN and SecRCNN (3000 iterations)

Fig. 7. Performance analysis of SecRCNN with RPN loss.



(a) R-CNN loss for Faster R-CNN and SecRCNN (3000 iterations)



(b) Computation error for R-CNN loss between Faster R-CNN and SecRCNN (3000 iterations)

Fig. 8. Performance analysis of SecRCNN with R-CNN loss.

and $q = 2^{13} - 1$. The Numpy and Tensorflow libraries of Python were utilized in the experiments to accelerate the parallel computations. In addition, the iteration numbers of `SDiv`, `SLog` and `SExp` were all set to be 50 by default. The learning rate was set to 0.001.

First, we compared the performance between Faster R-CNN and SecRCNN in terms of object detection accuracy and training loss. The comparison results are shown from Fig.6 to Fig.8. The loss of Faster R-CNN is composed of two parts, RPN loss and R-CNN loss, given in Fig. 7(a) and Fig. 8(a). From the experimental results, it can be discovered that SecRCNN attains similar performance with Faster R-CNN in the training process. The differences in the accuracy and the two types of losses between Faster R-CNN and SecRCNN are negligible. To further evaluate the computation errors, we sampled the exact error values at an interval of 500, as shown in Fig. 6(b), Fig. 7(b) and Fig. 8(b). In Fig. 6(b), the regression accuracy computation error for RPN was calculated by computing the average computation error of the four box bounding coefficients. As mentioned in Section V, the RPN loss and R-CNN loss can be split into two parts, satisfying RPN Loss = RPN-CLS Loss + RPN-REG Loss and RCNN Loss = RCNN-CLS Loss + RCNN-REG Loss, where RPN-CLS Loss, RPN-REG Loss, RCNN-CLS Loss and RCNN-REG Loss are the classification loss and the regression loss for RPN and RCNN. Therefore, we separately give the computation errors of the RPN loss and the R-CNN loss in Fig. 7(b) and Fig. 8(b). In the three graphs, it can be found that although the computation errors increase along with the training steps, the increasing rate is quite low. After 3000 iterations, the computation error increases by only one order of magnitude and maintains about 10^{-14}, which is completely negligible in real-world applications. The reason for the increase is that the computation error introduced in each round of training is accumulated along with the training process.

To experimentally investigate the security of SecRCNN, an example of medical image sharing is given in Fig.9. The six images in the first row of Fig.9 are from a Kaggle competition task[2] to find the nuclei in divergent images to advance medical discovery. The secretly shared images were generated by splitting the pixels of the original images into two random shares. The shared images are shown in the second and third rows of Fig.9. It can be found that the original images were transformed into two disorganized and meaningless images after being secretly shared. Intuitively, the image shares that look like noise images cannot reveal any information about the original images. By reflecting the random shares into gray values ranged from 0 and 255, the intuitive conclusion is further proved through the gray-level frequency histograms in Fig.10. For simplicity, we only selected the first two columns of Fig.9 to show their gray-level frequency in histograms. The sampling interval of the gray value is 2. It can be observed that for the original image, the gray values with high frequencies are intensively distributed in a specific area. Nevertheless, for the secretly shared images, their gray values are uniformly distributed between 0 to 255. The phenomenon means that the statistical distribution feature of the original medical image is hidden. Thus, it is quite hard for an attacker to exploit the image shares to infer any useful information about the original images. Furthermore, from the images in the last row of Fig.9, it can be discovered that the original images can be recovered by simply adding the corresponding random shares.

Then, we performed an analysis to evaluate the efficiency of SecRCNN. Table II illustrates the protocol runtime and message size of secure RPN and Fast R-CNN for training and testing. In each iteration, there were 128 feature maps pushed into the secure VGG-16 network. 17100 anchors for each map were generated according to the method described in Section V. Under the condition, SecRCNN can finish one iteration with about three minutes runtime and 150 millibyte communication overhead. Compared with model training, the testing needs much less computing resources and communication load. The reason is that for testing, there is no need to spend high computation resource on the computation of the

[2]https://www.kaggle.com/byrachonok/find-the-nuclei-in-divergent-medical-images/data

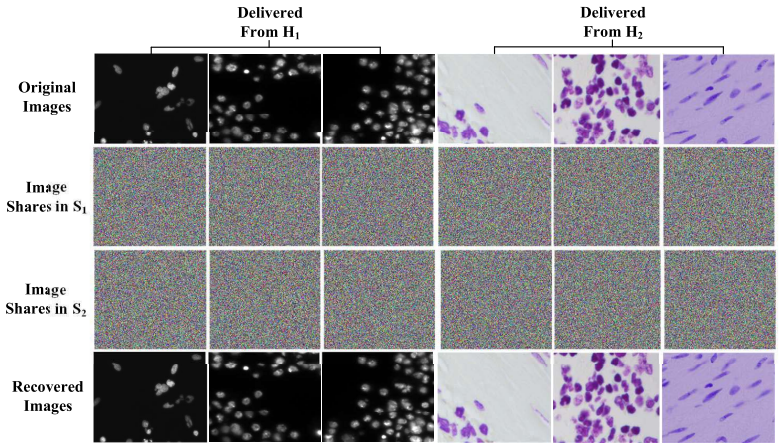


Fig. 9. Examples of medical image shares. First row: six medical images delivered by H_1 and H_2. Second row: the random shares of the first row for S_1. Third row: the random shares of the first row for S_2. Fourth row: the recovered images by adding the second row and the third row together.

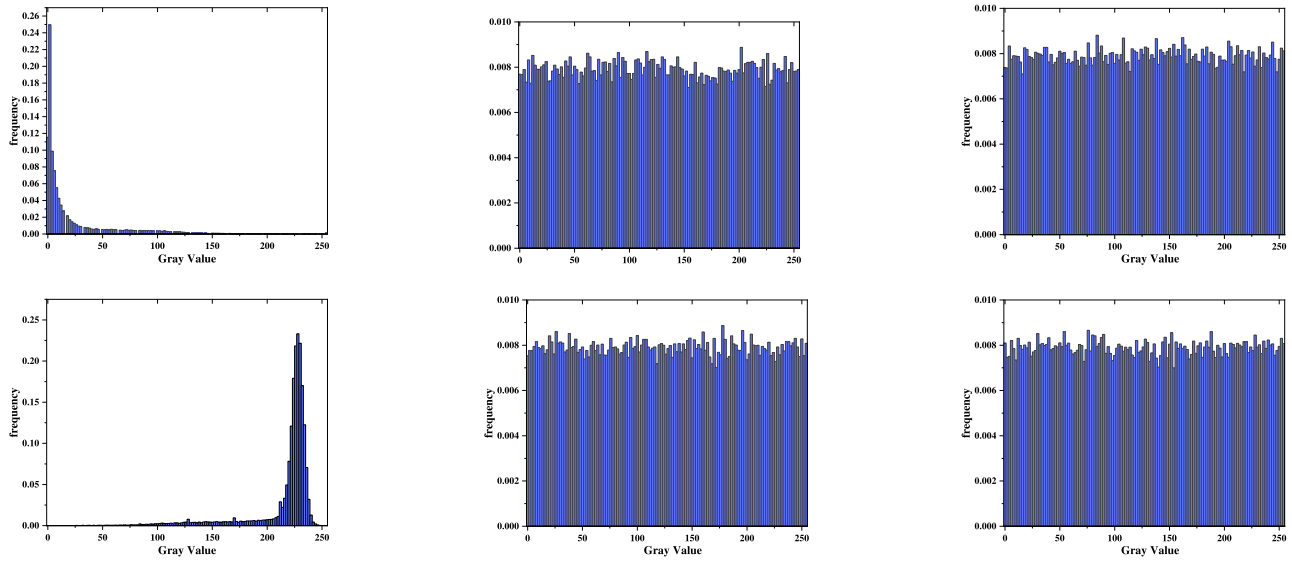


Fig. 10. Gray-level frequencies (histogram) for the first image from H_1 and the first image from H_2 in Fig.9. Left: the original images. Middle: the secretly shared images for S_1. Right: the secretly shared images for S_2.

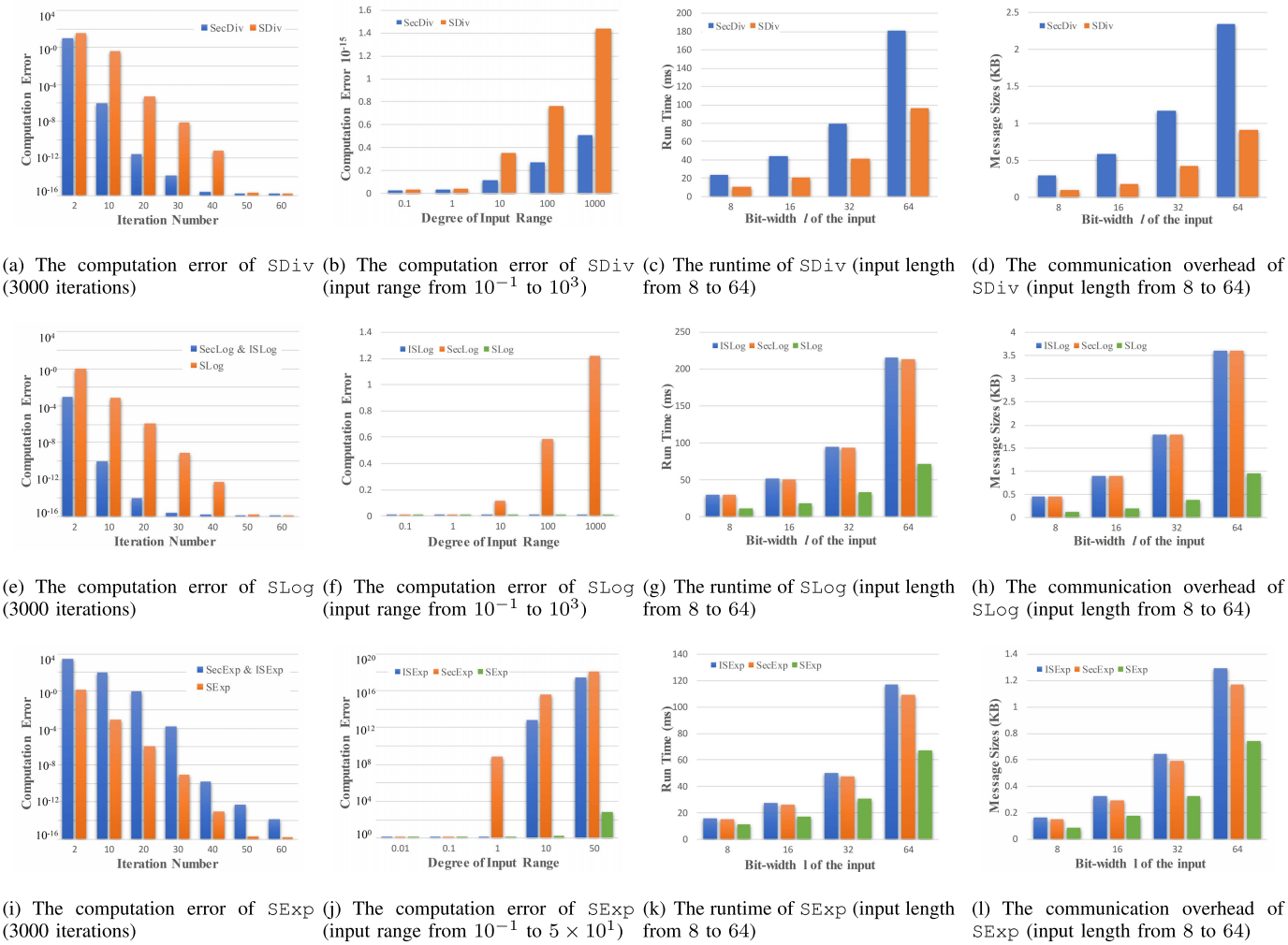
loss functions. In addition, assume that the size of input images is $w \times h$ and the data storage length is ℓ. After the feature extraction and region proposal stages, $\frac{w}{16} \times \frac{h}{16}$ features and $\frac{w}{16} \times \frac{h}{16} \times 9$ anchors are obtained. As mentioned in Section V, three sub-protocols are invoked in SRPN, which takes $6 \times 18 \times \frac{w}{16} \times \frac{h}{16} \times \ell$ bits communication overhead. Therefore, it theoretically requires $\mathcal{O}(wh\ell)$ bits to run SRPN. Then, suppose the mini-batch size to be $\mathcal{N}_{cls}$ and the number of anchors proposed by SPRN to be $\mathcal{N}_{reg}$. $\mathcal{O}((\mathcal{N}_{cls}+\mathcal{N}_{reg})\ell)$ bits are exchanged during the secure classification and bounding box regression stages.

B. Performance Evaluation of the Sub-protocols in SecRCNN

Four factors are important to evaluate the performance of the `SDiv`, `SLog` and `SExp`: 1) the computation errors with different iteration numbers; 2) the computation errors with different input ranges; 3) the runtime with different input lengths; and 4) the communication overhead with different input lengths. Consequently, we compared the performance of the sub-protocols proposed in Section IV with the previously proposed protocols [18]. In the graphs, `SecDiv`, `SecExp`, `ISEexp`, `SecLog` and `ISLog` are the abbreviations for the secure division, secure natural exponentiation, improved secure natural exponentiation, secure natural logarithm and improved secure natural logarithm sub-protocols, respectively. The experiment results of `SecLog` and `ISLog` are shown in the same column of the histograms because they are based on the identical iterative formula and have similar convergence features, as do `SecExp` and `ISExp`.

TABLE II
RUNTIME AND MESSAGE SIZE OF EACH ITERATION IN SECRCNN

Stage	RunTime (s)		Message Size (MB)	
	Training	Testing	Training	Testing
RPN Network	145.92	5.776	130.253	37.656
Fast R-CNN	46.47	1.811	27.515	6.289



(a) The computation error of `SDiv` (3000 iterations)
(b) The computation error of `SDiv` (input range from 10^{-1} to 10^{3})
(c) The runtime of `SDiv` (input length from 8 to 64)
(d) The communication overhead of `SDiv` (input length from 8 to 64)
(e) The computation error of `SLog` (3000 iterations)
(f) The computation error of `SLog` (input range from 10^{-1} to 10^{3})
(g) The runtime of `SLog` (input length from 8 to 64)
(h) The communication overhead of `SLog` (input length from 8 to 64)
(i) The computation error of `SExp` (3000 iterations)
(j) The computation error of `SExp` (input range from 10^{-1} to 5×10^{1})
(k) The runtime of `SExp` (input length from 8 to 64)
(l) The communication overhead of `SExp` (input length from 8 to 64)

Fig. 11. Effectiveness and efficiency analysis of sub-protocols in SecRCNN.

From Fig. 11(a), Fig. 11(e) and Fig. 11(i), it can be observed that our sub-protocols converge slightly slower than the existing protocols that are Maclaurin Series and Newton-Raphson based [18]. However, they can still reach 10^{-10} computation error after about 30 iterations, which is enough to satisfy most applications. When the input range changes as shown in Fig. 11(b), Fig. 11(f) and Fig. 11(j), all the newly proposed sub-protocols can at least reach or even exceed the performance of the existing sub-protocols. As shown in Fig. 11(c), Fig. 11(g) and Fig. 11(k), the runtime needed by `SDiv`, `SLog` or `SExp` is obviously less than the previous secure sub-protocols. The decrease is because the messages exchanged in our CORDIC based protocols are greatly reduced, listed in Fig. 11(d), Fig. 11(h) and Fig. 11(l). Data communication is the most time-consuming operation in the additive secret sharing based framework.

VIII. Related Work

Faster R-CNN is one of the most successful convolutional networks evolved from R-CNN [11], which is the first type of deep learning network applied to the domain of object detection. In recent years, the model has been widely deployed in the fields of autonomous driving [26], biometric authentication [27], and especially medical diagnosis [28]. Due to the state-of-the-art performance of Faster R-CNN, Zhang *et al.* [29] succeeded in identifying and detecting the adhesion cancer cells in phase-contrast microscopy with limited samples, which was essential to help people against cancer. Later, Ding *et al.* [30] made it possible to accurately detect pulmonary nodules with computed tomography images for the early screening of lung cancer using Faster R-CNN. Besides cancer, Faster R-CNN also has great clinical significance in the treatment of other diseases. Lo *et al.* [31] utilized Faster R-CNN to detect the glomeruli in light microscopic images of renal pathology, which is significant for the diagnosis of kidney diseases To reduce the difficulty of examining the transthalamic plane of the fetal head, Lin *et al.* [32] presented a Faster R-CNN based automatic method to assess the quality of the fetal head in ultrasound images. In addition, by combining Faster R-CNN with DeepLab [33], Tang *et al.* [34] overcame the challenging task of segmenting the liver from other organ tissues in clinical images and achieved automatic liver segmentation.

To guarantee the privacy of patients' data while analyzing their pathological images, the most popular solution is digital watermarking. By inserting the binary format watermarking information into the pixel value of the original image, the

approach can protect the authenticity and integrity of the medical images. For example, Selvaraj and Varatharajan [36] used the hash function to improve the ability of watermarking to protect the integrity of digital medical images. However, in general, the quality of the watermarked images is reduced to a certain extent [36]. In a situation where multiple healthcare centers would like to cooperate to build a deep learning model, watermarking is no longer suitable. This is because the watermarked images are not computable any more. And since watermark can only provide authenticity of patient ID, it cannot hide the image information from being stolen to build an anonymous database by others. To tackle this problem, Wu [37] utilized the HE to provide reversible hiding of medical images. CaRENets proposed by Chao *et al.* [38] was also based on the HE and support the inference of tge CNN model on the encrypted medical images. However, because of the high requirements for computation power and storage space, HE is not practical in applications. The other approach for privacy preservation of medical images is called garbled-circuit. Zheng *et al.* [9] used GC to protect the medical image privacy from the external cloud database. Unfortunately, GC is also unpractical due to the same reason as HE. In addition, the differential privacy (DP) technique can also be used to protect the privacy of medical images [39]. DP is an approach that is specially designed to preserve data privacy while applying the data mining technique. By adding random noises to the image pixels, DP makes a single image unrecognizable for the adversary, but the statistical feature for the whole database is still reserved. Nevertheless, since there has not been a novel algorithm, the existing DP based schemes always make the deep learning model suffer from a dramatic reduction in accuracy [10].

IX. Conclusion

In this paper, we proposed SecRCNN, a privacy-preserving object detection model for medical image analysis. To reduce the communication overhead of the iterative approximation process, we redesigned the existing sub-protocols of common mathematical functions with the CORDIC algorithms. Then, we proposed a series of interactive protocols to implement the training and inference process of SecRCNN, which included feature extraction, region proposal, classification and bounding box regression. Based on SecRCNN, healthcare centers can collaborate to train a more accurate and more reliable model without concern of privacy disclosure.

Appendix

A. An Example of Image Data Split

To illustrate the medical image split process of SecRCNN, we give a simple example as shown in Fig. 12. In the example, the original medical image pixels are given as a very simple grey-value matrix. For RGB images or other types of images, the difference between them and the grey-value image is that each pixel is represented with more dimensions. Thus, we can repeat the operations of the example to complete the image split of them. The following is the description of the example. The image in the example is owned by H_1. The representation

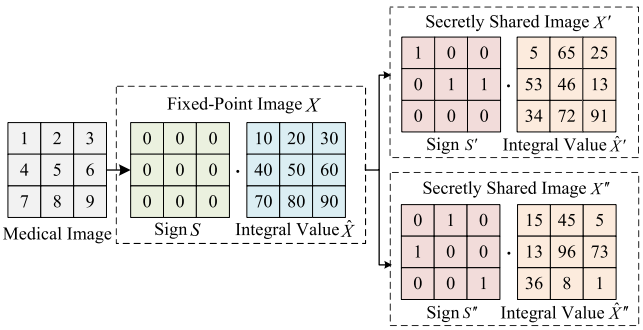


Fig. 12. A simple example of medical image split with $p = 101$ and representation precision $c = 1$.

precision c is set to 1. To split the image into random shares, H_1 first transforms the grey-value matrix into the fixed-point format $X = (-1)^{\mathrm{S}} \cdot \hat{X} \cdot 10^{-1}$. Then, select a uniformly random value for each pixel and merge the values into $\hat{X}'$. Meanwhile, corresponding to $\hat{X}'$, randomly select the sign matrix S'. Next, compute $\hat{X}_0 = (-1)^{\mathrm{S}} \cdot \hat{X} \cdot 10^{-1} - (-1)^{\mathrm{S}'} \cdot \hat{X}' \cdot 10^{-1}$ and $\hat{X}'' = |\hat{X}_0| \mod p$. S'' corresponds to the sign of $\hat{X}_0$. Finally, H_1 can get the random shares of the medical image, $X' = (-1)^{\mathrm{S}'} \cdot \hat{X}' \cdot 10^{-1}$ and $X'' = (-1)^{\mathrm{S}''} \cdot \hat{X}'' \cdot 10^{-1}$, which satisfy $X = X' + X''$.

B. Additive Secret Sharing Protocols

The implementation details of three previously proposed additive secret sharing sub-protocols [15], `SAdd`, `SMul` and `SCmp` are given in Protocol 9, Protocol 10 and Protocol 11.

Protocol 9 Secure Addition Protocol `SAdd`

Input: S_1 has input μ_1, v_1; S_2 has input μ_2, v_2.
Output: S_1 outputs f_1; S_2 outputs f_2;
1: S_1 computes $f_1 = \mu_1 + v_1$.
2: S_2 computes $f_2 = \mu_2 + v_2$.

Protocol 10 Secure Multiplication Protocol `SMul`

Input: S_1 has input μ_1, v_1; S_2 has input μ_2, v_2.
Output: S_1 outputs f_1; S_2 outputs f_2;
1: $\mathcal{T}$ generates random numbers a, b and computes $c = a \cdot b$.
2: $\mathcal{T}$ splits a, b, and c into random shares: $a = a_1 + a_2$, $b = b_1 + b_2$, and $c = c_1 + c_2$.
3: $\mathcal{T}$ sends a_i, b_i, and c_i to S_i $(i = 1, 2)$.
4: S_1 computes $\alpha_1 \leftarrow \mu_1 - a_1$, $\beta_1 \leftarrow v_1 - b_1$, and sends α_1, β_1 to S_2.
5: S_2 computes $\alpha_2 \leftarrow \mu_2 - a_2$, $\beta_2 \leftarrow v_2 - b_2$, and sends α_2, β_2 to S_1.
6: S_1 computes $\alpha \leftarrow \alpha_1 + \alpha_2$, $\beta \leftarrow \beta_1 + \beta_2$, and $f_1 \leftarrow c_1 + b_1 \cdot \alpha + a_1 \cdot \beta$.
7: S_2 computes $\alpha \leftarrow \alpha_1 + \alpha_2$, $\beta \leftarrow \beta_1 + \beta_2$, and $f_2 \leftarrow c_2 + b_2 \cdot \alpha + a_2 \cdot \beta + \alpha \cdot \beta$.

C. An Example of `SDiv`

To better explain the workflow of our secure protocols, we introduce an example of `SDiv`, shown in Fig. 13.

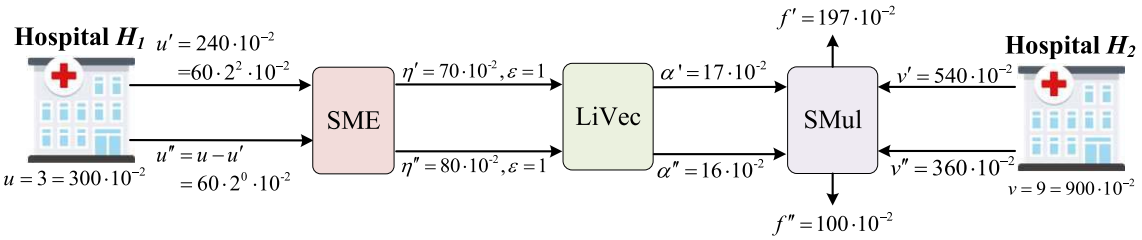


Fig. 13. An example of the secure division protocol for computing $f' + f'' = \upsilon/\mu = 9/3 = 3$

Protocol 11 Secure Comparison Protocol SCmp

Input: S_1 has input μ_1' and μ_1''; S_2 has input μ_2' and μ_2''; The data length of μ_1 and μ_2 is l.

Output: S_1 outputs f_1; S_2 outputs f_2;

1: For $j = 0, \ldots, l-1$, $\mathcal{T}$ generates random $r_1^{(j)}$, $r_2^{(j)}$.
2: For $j = 0, \ldots, l-1$, $\mathcal{T}$ generates random $r \leftarrow r_1^{(j)} \oplus r_2^{(j)}$.
3: $\mathcal{T}$ computes $r \leftarrow -r^{(l-1)} \cdot 2^{l-1} + \sum_{j=0}^{l-2} r^{(j)} \cdot 2^j$.
4: $\mathcal{T}$ generates random s_1 and computes $s_2 \leftarrow r - s_1$.
5: $\mathcal{T}$ sends s_i and $r_i^{(l-1)} \ldots r_i^{(0)}$ to $\mathcal{S}_i$ $(i = 1, 2)$.
6: $\mathcal{S}_i$ compute $\mu_i = \mu_i' + \mu_i''$
7: S_1 computes $t_1 \leftarrow \mu_1 - s_1$.
8: S_2 computes $t_2 \leftarrow \mu_2 - s_2$ and sends it to S_1.
9: S_1 computes $\upsilon \leftarrow t_1 + t_2$ and generates their complement binary representation $\upsilon^{(l-1)} \ldots \upsilon^{(0)}$.
10: For $j = 0, \ldots, l-1$, S_1 generates random $v_1^{(j)}$ and computes $v_2^{(j)} \leftarrow \upsilon^{(j)} \oplus v_1^{(j)}$.
11: S_1 sends $v_2^{(l-1)} \ldots v_2^{(0)}$ to S_2.
12: **for** $j = 0$ to $l-1$ **do**
13: S_1 computes $\alpha_i^{(j)} \leftarrow v_i^{(j)} \oplus r_i^{(j)}$.
14: S_1 and S_2 compute $(\beta_1^{(j)}, \beta_2^{(j)}) \leftarrow \mathtt{SMul}(v_1^{(j)}, v_2^{(j)}, r_1^{(j)}, r_2^{(j)})$.
15: **end for**
16: $\mathcal{S}_i$ set $c_i^{(0)} \leftarrow 0$.
17: $\mathcal{S}_i$ computes $u_i^{(0)} \leftarrow v_i^{(0)} \oplus r_i^{(0)}$.
18: **for** $j = 1$ to $l-1$ **do**
19: S_1 and S_2 compute $(\alpha_1^{(j-1)}, \alpha_2^{(j-1)}) \leftarrow \mathtt{SMul}(\alpha_1^{(j-1)}, \alpha_2^{(j-1)}, c_1^{(j-1)}, c_2^{(j-1)})$.
20: $\mathcal{S}_i$ compute $c_i^{(j)} \leftarrow \alpha_i^{(j-1)} \oplus \beta_i^{(j-1)}$.
21: $\mathcal{S}_i$ compute $u_i^{(j)} \leftarrow v_i^{(j)} \oplus r_i^{(j)} \oplus c_i^{(j)}$.
22: **end for**
23: $\mathcal{S}_i$ return $f_i = u_i^{l-1}$.

In the example, we set $q = 1009$, $p = 8191$ and $c = 2$. The original data are $\mu = 3 = 300 \cdot 10^{-2}$ and $\upsilon = 9 = 900 \cdot 10^{-2}$, which are owned by healthcare centers H_1 and H_2, respectively. To split μ into random shares, H_1 first selects a uniformly random value $\hat{\mu}' = 240$ from Z_{8191} and $\mathtt{s}' = 0$ from $\{0, 1\}$. Then, compute $\mu - (-1)^{\mathtt{s}'} \cdot \hat{\mu}' \cdot 10^{-2} = 0.6$ and obtain $\mathtt{s}'' = 0$, $\hat{\mu}'' = 60$. Finally, H_1 has $\mu' = (-1)^0 \cdot 240 \cdot 10^{-2}$ and $\mu'' = (-1)^0 \cdot 60 \cdot 10^{-2}$, the two random shares of μ. In the same way, H_2 splits υ. The shares are uploaded to the two edge servers S_1 and S_2. S_1 and S_2 orderly operate the three sub-protocols of SDiv and get the outputs $f' = 197 \cdot 10^{-2} = 1.97$ and $f'' = 100 \cdot 10^{-2} = 1.00$, respectively. f' and f'' satisfy $f = f' + f'' = \upsilon/\mu = 2.97$. The computation error is caused by the compulsive approximation of fixed-point representation. The larger c is, the smaller the computation error is.

REFERENCES

[1] G. Litjens *et al.*, "A survey on deep learning in medical image analysis," *Med. Image Anal.*, vol. 42, pp. 60–88, Dec. 2017.
[2] J. Ker *et al.*, "Deep learning applications in medical image analysis," *IEEE Access*, vol. 6, pp. 9375–9389, 2017.
[3] J. H. Lee and K. G. Kim, "Applying deep learning in medical images: The case of bone age estimation," *Healthcare Inform. Res.*, vol. 24, no. 1, p. 86–92, 2018.
[4] S. K. Pandey and R. R. Janghel, "Recent deep learning techniques, challenges and its applications for medical healthcare system: A review," *Neural Process. Lett.*, no. 1, pp. 1–29, Jan. 2019.
[5] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.
[6] A. Krishnan and M. L. Das, "Medical image security with cheater identification using secret sharing scheme," in *Proc. Int. Conf. Signal, Netw., Comput., Syst.*, 2017, pp. 117–126.
[7] Y. Yang, W. Zhang, D. Liang, and N. Yu, "A ROI-based high capacity reversible data hiding scheme with contrast enhancement for medical images," *Multimedia Tools Appl.*, vol. 77, no. 14, pp. 18043–18065, 2018.
[8] L. Wang, L. Li, L. Jin, L. Jing, B. B. Gupta, and L. Xia, "Compressive sensing of medical images with confidentially homomorphic aggregations," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1402–1409, Apr. 2019.
[9] Y. Zheng, H. Cui, C. Wang, and J. Zhou, "Privacy-preserving image denoising from external cloud databases," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 6, pp. 1285–1298, Jun. 2017.
[10] M. Abadi *et al.*, "Deep learning with differential privacy," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2016, pp. 308–318.
[11] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2015, pp. 91–99.
[12] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: https://arxiv.org/abs/1409.1556
[13] P. Belanović and M. Leeser, "A library of parameterized floating-point modules and their use," in *Proc. Int. Conf. Field Programm. Logic Appl.* Berlin, Germany: Springer, 2002, pp. 657–666.
[14] D. Bogdanov, M. Niitsoo, T. Toft, and J. Willemson, "High-performance secure multi-party computation for data mining applications," *Int. J. Inf. Secur.*, vol. 11, no. 6, pp. 403–418, Nov. 2012.
[15] K. Huang, X. Liu, S. Fu, D. Guo, and M. Xu, "A lightweight privacy-preserving CNN feature extraction framework for mobile sensing," *IEEE Trans. Dependable Secure Comput.*, to be published.
[16] X. Liu, R. H. Deng, K.-K. R. Choo, and J. Weng, "An efficient privacy-preserving outsourced calculation toolkit with multiple keys," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 11, pp. 2401–2414, Nov. 2016.
[17] X. Liu, R. Deng, K.-K. R. Choo, and Y. Yang, "Privacy-preserving reinforcement learning design for patient-centric dynamic treatment regimes," *IEEE Trans. Emerg. Topics Comput.*, to be published.
[18] Z. Ma, Y. Liu, X. Liu, J. Ma, and K. Ren, "Lightweight privacy-preserving ensemble classification for face recognition," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 5778–5790, Jun. 2019.
[19] J. E. Volder, "The CORDIC trigonometric computing technique," *IRE Trans. Electron. Comput.*, vol. EC-8, no. 3, pp. 330–334, 1959.

[20] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

[21] I. Damgård, M. Fitzi, E. Kiltz, J. B. Nielsen, and T. Toft, "Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation," in *Proc. Theory Cryptogr. Conf.* Berlin, Germany: Springer, 2006, pp. 285–304.

[22] D. Beaver, "Efficient multiparty protocols using circuit randomization," in *Proc. Annu. Int. Cryptol. Conf.* Berlin, Germany: Springer, 1991, pp. 420–432.

[23] D. Bogdanov, S. Laur, and J. Willemson, "Sharemind: A framework for fast privacy-preserving computations," in *Proc. Eur. Symp. Res. Comput. Secur.* Berlin, Germany: Springer, 2008, pp. 192–206.

[24] I. Dimitrovski, D. Kocev, S. Loskovska, and S. Džeroski, "Hierarchical annotation of medical images," *Pattern Recognit.*, vol. 44, nos. 10–11, pp. 2436–2449, 2011.

[25] D. Demmler, G. Dessouky, F. Koushanfar, A.-R. Sadeghi, T. Schneider, and S. Zeitouni, "Automated synthesis of optimized circuits for secure computation," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, 2015, pp. 1504–1517.

[26] Q. Fan, L. Brown, and J. Smith, "A closer look at faster R-CNN for vehicle detection," in *Proc. IEEE Intell. vehicles Symp. (IV)*, Jun. 2016, pp. 124–129.

[27] H. Jiang and E. Learned-Miller, "Face detection with the faster R-CNN," in *Proc. 12th IEEE Int. Conf. Autom. Face Gesture Recognit. (FG)*, May/Jun. 2017, pp. 650–657.

[28] J. Hung and A. Carpenter, "Applying faster R-CNN for object detection on malaria images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, Jul. 2017, pp. 56–61.

[29] J. Zhang, H. Hu, S. Chen, Y. Huang, and Q. Guan, "Cancer cells detection in phase-contrast microscopy images based on faster R-CNN," in *Proc. IEEE 9th Int. Symp. Comput. Intell. Design (ISCID)*, vol. 1, Dec. 2016, pp. 363–367.

[30] J. Ding, A. Li, Z. Hu, and L. Wang, "Accurate pulmonary nodule detection in computed tomography images using deep convolutional neural networks," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assisted Intervent.* Cham, Switzerland: Springer, 2017, pp. 559–567.

[31] Y.-C. Lo *et al.*, "Glomerulus detection on light microscopic images of renal pathology with the faster R-CNN," in *Proc. Int. Conf. Neural Inf. Process.* Cham, Switzerland: Springer, 2018, pp. 369–377.

[32] Z. Lin *et al.*, "Quality assessment of fetal head ultrasound images based on faster R-CNN," in *Simulation, Image Processing, and Ultrasound Systems for Assisted Diagnosis and Navigation*. Cham, Switzerland: Springer, 2018, pp. 38–46.

[33] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Apr. 2017.

[34] W. Tang, D. Zou, S. Yang, and J. Shi, "DSL: Automatic liver segmentation with faster R-CNN and deeplab," in *Proc. Int. Conf. Artif. Neural Netw.* Cham, Switzerland: Springer, 2018, pp. 137–147.

[35] P. Selvaraj and R. Varatharajan, "Whirlpool algorithm with hash function based watermarking algorithm for the secured transmission of digital medical images," *Mobile Netw. Appl.*, pp. 1–14, May 2018.

[36] S. M. Mousavi, A. Naghsh, and S. Abu-Bakar, "Watermarking techniques used in medical images: A survey," *J. Digit. Imag.*, vol. 27, no. 6, pp. 714–729, 2014.

[37] X. Wu, "An algorithm for reversible information hiding of encrypted medical images in homomorphic encrypted domain," *Discrete Continuous Dyn. Syst.-S*, vol. 12, pp. 144–155, Aug. 2018.

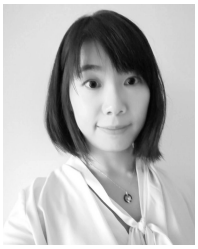
[38] J. Chao *et al.*, "CaRENets: Compact and resource-efficient CNN for homomorphic inference on encrypted medical images," 2019, *arXiv:1901.10074*. [Online]. Available: https://arxiv.org/abs/1901.10074

[39] M. Noura, H. Noura, A. Chehab, M. M. Mansour, L. Sleem, and R. Couturier, "A dynamic approach for a lightweight and secure cipher for medical images," *Multimedia Tools Appl.*, vol. 77, no. 23, pp. 31397–31426, 2018.

Yang Liu received the B.S. degree in computer science and technology from Xidian University in 2017, where he is currently pursuing the master's degree with the School of Cyber Engineering. His research interests cover formal analysis of protocols and deep learning neural network in cyber security.

Zhuo Ma (Member, IEEE) received the Ph.D. degree in computer architecture from Xidian University, Xi'an, China, in 2010. He is currently an Associate Professor with the School of Cyber Engineering, Xidian University. His research interests include cryptography, machine learning in cyber security, and the Internet of things security.

Ximeng Liu (Member, IEEE) received the B.Sc. degree in electronic engineering and the Ph.D. degree in cryptography from Xidian University, Xi'an, China, in 2010 and 2015, respectively. He is a currently a Full Professor with the College of Mathematics and Computer Science, Fuzhou University, China. He is also a Research Fellow with the School of Information System, Singapore Management University, Singapore. He has published over 100 research articles, including IEEE TIFS, IEEE TDSC, IEEE TC, IEEE TII, IEEE TSC, and IEEE TCC. His research interests include cloud security, applied cryptography, and big data security.



Siqi Ma received the B.S. degree in computer science and technology from Xidian University and the Ph.D. degree in information system from Singapore Management University. She is currently a Postdoctoral Research Fellow with Data 61, CSIRO. Her research interests include vulnerability detection and repair on software.

Kui Ren (Fellow, IEEE) is currently a Distinguished Professor with the School of Computer Science and Technology, Zhejiang University, where he also directs the Institute of Cyber Space Research. His research interests include cloud and data security, AI and IoT security, and privacy-enhancing technologies. He is also a Distinguished Member of ACM.