Toward Lightweight, Privacy-Preserving Cooperative Object Classification for Connected Autonomous Vehicles

Jinbo Xiong, *Member, IEEE*, Renwan Bi, Youliang Tian, *Member, IEEE*, Ximeng Liu, *Senior Member, IEEE*, and Dapeng Wu, *Senior Member, IEEE*

***Abstract*—Collaborative perception enables autonomous vehicles to exchange sensor data among each other to achieve cooperative object classification, which is considered an effective means to improve the perception accuracy of connected autonomous vehicles (CAVs). To protect information privacy in cooperative perception, we propose a lightweight, privacy-preserving cooperative object classification framework that allows CAVs to exchange raw sensor data (e.g., images captured by HD camera), without leaking private information. Leveraging chaotic encryption and additive secret sharing technique, image data are first encrypted into two ciphertexts and processed, in the encrypted format, by two separate edge servers. The use of chaotic mapping can avoid information leakage during data uploading. The encrypted images are then processed by the proposed privacy-preserving convolutional neural network (P-CNN) model embedded in the designed secure computing protocols. Finally, the processed results are combined/decrypted on the receiving vehicles to realize cooperative object classification. We formally prove the correctness and security of the proposed framework and carry out intensive experiments to evaluate its performance. The experimental results indicate that P-CNN offers exactly almost the same object classification results as the original CNN model, while offering great privacy protection of shared data and lightweight execution efficiency.**

***Index Terms*—Connected and autonomous vehicle, convolutional neural network (CNN), edge computing, object classification, privacy protection.**

Manuscript received April 1, 2021; revised May 24, 2021; accepted June 15, 2021. Date of publication June 30, 2021; date of current version February 4, 2022. This work was supported in part by the National Natural Science Foundation of China under Grant 61872088, Grant 61772008, Grant 62072109, Grant 61872090, Grant U1905211, Grant U1804263, and Grant U1836205; in part by the Natural Science Foundation of Fujian Province under Grant 2019J01276; in part by the Science and Technology Major Support Program of Guizhou Province under Grant 20183001; in part by the Science and Technology Program of Guizhou Province under Grant 20191098; in part by the Guangxi Key Laboratory of Cryptography and Information Security under Grant GCIS202105; and in part by the Project of High-level Innovative Talents of Guizhou Province under Grant 20206008. *(Corresponding author: Youliang Tian.)*

Jinbo Xiong and Renwan Bi are with the Fujian Provincial Key Laboratory of Network Security and Cryptology, College of Computer and Cyber Security, Fujian Normal University, Fuzhou 350117, China, and also with the Guangxi Key Laboratory of Cryptography and Information Security, Guilin University of Electronic Technology, Guilin 541004, China (e-mail: jbxiong@fjnu.edu.cn; brw2806@163.com).

Youliang Tian is with the State Key Laboratory of Public Big Data, College of Computer Science and Technology, Guizhou University, Guiyang 550025, China (e-mail: youliangtian@163.com).

Ximeng Liu is with the Key Laboratory of Information Security of Network Systems, Fuzhou University, Fuzhou 350108, China (e-mail: snbnix@gmail.com).

Dapeng Wu is with the Chongqing Key Laboratory of Optical Communication and Networks, Chongqing University of Posts and Telecommunications, Chongqing 400065, China (e-mail: wudapengphd@gmail.com).

Digital Object Identifier 10.1109/JIOT.2021.3093573

I. Introduction

THANKS to the perfect combination of foremost wireless communication and autonomous vehicle technologies, a connected autonomous vehicle (CAV) system plays an important role in facilitating our daily life, e.g., accomplishing real-time tasks, talking with the business, and working in a mobile environment [1]. Typically, in a CAV system, each autonomous vehicle leverages a suite of local sensors [e.g., high-definition (HD) cameras] to perceive its surrounding environment [2], [3]. The perception results, e.g., nearby objects' locations (and types), can be shared among vehicles to realize cooperative perception [4], and achieve a CAV system.

A. Problem Statement

It has been discovered recently that sharing raw sensor data, instead of processed ones, among autonomous vehicles could significantly improve the performance of object detection in a CAV system [5]. The basic idea is to let edge servers collect the raw sensor data, e.g., produced by HD camera sensors, from nearby autonomous vehicles to conduct more accurate cooperative object detection [6]. Sharing raw perception data among vehicles or between vehicles and edge servers, however, faces serious privacy leakage issues, due to the fact that the captured data contain a large amount of sensitive information [7], [8]. For example, image data captured by HD cameras may contain drivers' or pedestrians' faces, licence plates, roadside buildings, and location information [9]. How to protect these sensitive information will bring great challenges to the CAV system [10]. The existing data privacy protection methods, such as anonymity, obfuscation, and cryptography, will create serious problems, reducing data availability and identification accuracy, and increasing tremendous computational cost. Therefore, it is a profound issue to design a lightweight, privacy-preserving cooperative object detection/classification mechanism for CAVs.

B. Limitation of Prior Arts

Convolutional neural network (CNN) [11] is a standard art for image classification. However, training a high accuracy CNN model for the CAV system is usually very complex, requiring huge computational cost and storage capacity. Therefore, due to the computation-intensive nature of data processing based on CNN for the CAV system, cooperative perception among vehicles is usually implemented by offloading complex computational tasks from autonomous vehicles to edge servers [12], [13]. This is particularly true when sensor data from several vehicles are fused by the edge servers to realize a cooperative object detection [14]. As information is leaving capturing vehicles, privacy protection of captured sensor data must be placed in order to take the full advantage of edge computing [15], [16].

In this regard, sending raw data directly to edge server for processing is obviously impractical and insecure [17]. Majority of existing security solutions leverage data encryption techniques [18], i.e., sensor data are encrypted before transmission, and decrypted and processed after reception. A sending vehicle encrypts its sensor data and transmits it to a receiving vehicle in the ciphertext to ensure data security during the transmission process. It is still possible for privacy leakage after the data are decrypted on receivers/edge servers. To address this issue, homomorphic encryption (HE) [19] is undoubtedly the best choice because data processing can be carried out over ciphertexts, without the need of knowing plaintext information. CryptoNets [20] and CryptoDL [21] employed additively HE (AHE) to realize simple computation tasks involved in neural networks. Later, Juvekar *et al.* [22] used AHE and garbled circuit-based two-party computation to improve computing efficiency to some extent. Unfortunately, the computational cost and processing latency brought by HE are still overburdened for CAV applications. More importantly, the raw perception data are fully exposed to the receiving vehicle, and once the vehicle is compromised, the consequences are terrible.

C. Proposed Solution

In order to solve the above-mentioned problems, we employ a more lightweight additive secret sharing (ASS) scheme to implement the computational operations involved in a CNN, using a two-tier edge architecture [23], [24], to offload heavy object classification tasks from autonomous vehicles to two edge servers. In this way, edge servers are able to process encrypted images by using the proposed privacy-preserving CNN (P-CNN) to accomplish cooperative object classification.

To ensure the secure storage and transmission of image data [25], it is necessary to implement an effective image encryption processing. As such, a sending vehicle randomly splits the original image, captured by HD cameras on autonomous vehicles, into two image shares with the same dimension. Then, these two shares are encrypted and transmitted to two edge servers using chaotic maps to avoid information leakage caused by simultaneous channel attacks. These edge servers conduct privacy-preserving operations within a CNN to discover and detect objects on encrypted image shares, and collaboratively perform the cooperative object classification task. In the entire process, image privacy is always protected; in addition, the computational cost and communication overhead are offloaded to edge servers.

D. Contributions

The major contributions of this article can be summarized as follows.

1) We propose a lightweight, privacy-preserving framework that is specifically designed to realize a cooperative object classification in CAV systems. In the framework, image reencryption, privacy-preserving CNN, and object decryption are aiming to achieve secure and efficient cooperative object classification for CAVs. This is a generic framework since it can be extended to other CNN-based deep learning models, as long as the computation tasks involved in these models are equivalently implemented by the proposed secure computing protocols.
2) Before the raw perception data are uploaded to the edge server, an image chaotic encryption scheme is designed to prevent potential adversaries from hijacking sending vehicles or simultaneous damaging both upload links. If both links are compromised by the adversary, the original images can be easily recovered by addition operation since they are randomly split by ASS.
3) A series of secure computing protocols is designed in P-CNN to (almost) identically implement the computational operations associated with the original CNN. Moreover, these protocols are well suited for multidimensional arrays. As a result, the computational efficiency of P-CNN is greatly improved, when compared to the elementwise calculations.
4) As one of the heaviest computational tasks, encrypted image-based processing is carried out on edge servers. Therefore, the computational cost on individual vehicle is significantly reduced. CAVs, in contrast, only perform simple addition calculations. Thorough theoretical analysis proves the security and correctness of the proposed protocols and framework. Comprehensive experimental results further verify that our framework is superior to the existing schemes in terms of computational cost and communication overhead.

The remainder of this article is organized as follows. In Section II, we introduce the necessary preliminaries. The system architecture and security model are presented in Section III. In Section IV, we discuss in detail the implementation process of the proposed framework, and describe some secure computing protocols involved in P-CNN based on the ASS. Theoretical analysis and experimental results are presented in Sections V and VI, respectively. Section VII introduces the related work, and Section VIII summarizes the article.

II. Preliminaries

In this section, we introduce some necessary knowledge, including chaos map, CNN model, and ASS.

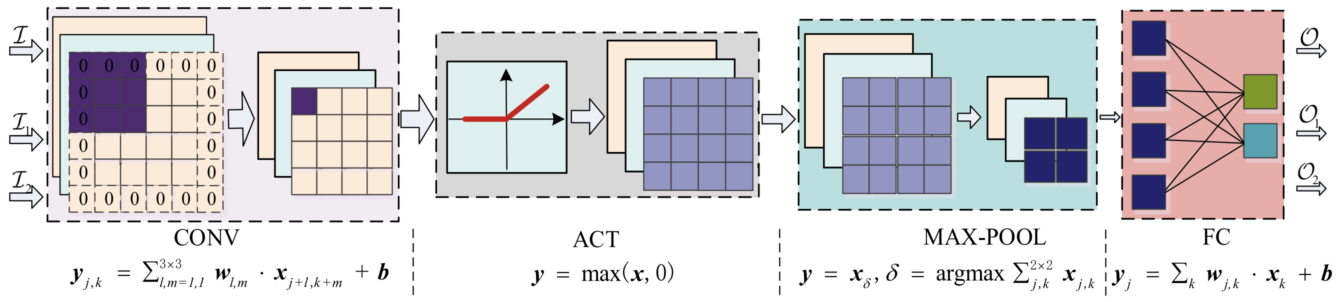
$\mathcal{I}$
$\mathcal{I}_1$
$\mathcal{I}_2$
$\mathcal{O}$
$\mathcal{O}_1$
$\mathcal{O}_2$
CONV
$y_{j,k} = \sum_{l,m=1,1}^{3\times3} w_{l,m} \cdot x_{j+l,k+m} + b$
ACT
$y = \max(x, 0)$
MAX-POOL
$y = x_\delta, \delta = \text{argmax} \sum_{j,k}^{2\times2} x_{j,k}$
FC
$y_j = \sum_k w_{j,k} \cdot x_k + b$

Fig. 1. Major operations in the CNN model, followed by the corresponding functions.

A. *Chaotic Map*

Chaos [26] is a phenomenon that appears to be random and irregular motion in a certain system. The map sequence based on the chaos theory is very sensitive to the control parameters and initial values. This statistical characteristics can be used to design image diffusion and obfuscation schemes. Moreover, compared with image iterative encryption based on symmetric keys, the chaotic encryption method has lower computational cost and is more consistent with image encryption and transmission in the context of CAV.

Here, we introduce two discrete chaotic maps: 1) logistic map [27] and 2) lorenz map [28]. The logistic map is a classical 1-D discrete map, which is defined as $\boldsymbol{\phi}(k+1) = \varepsilon\boldsymbol{\phi}(k)(1-\boldsymbol{\phi}(k))$, where ε is the control parameter. When $3.57 < \varepsilon \leq 4$, the map is chaotic. The lorenz map is a 3-D discrete map, which is defined as

$$\begin{cases} \boldsymbol{\xi}_1(k+1) = \boldsymbol{\xi}_1(k)\boldsymbol{\xi}_2(k) - \boldsymbol{\xi}_3(k) \\ \boldsymbol{\xi}_2(k+1) = \boldsymbol{\xi}_1(k) \\ \boldsymbol{\xi}_3(k+1) = \boldsymbol{\xi}_2(k). \end{cases} \tag{1}$$

It is worth noting that the image encryption scheme based on single map is slightly weak in security strength. Obviously, it is reasonable to combine the above two maps to design a higher key-dimensional image encryption scheme without significantly increasing computational cost. See Section IV-A1 for the construction process.

B. *CNN Model*

The CNN model widely used for object classification tasks usually contains several convolutional (CONV) layers, activation (ACT) layers, pooling (POOL) layers, and full-connected (FC) layers, as shown in Fig. 1. Below, we introduce the major functions and operations involved in the four types of layers.

For the CONV layer, the main purpose is to extract local features from images. Essentially, the underlying calculation can be viewed as a linear combination of the data from images and the convolutional core. Assuming the input and output is $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively, we denote $\boldsymbol{y}_{j,k} = \Sigma_{m',m''=1,1}^{C,C} \boldsymbol{x}_{j+m',k+m''} \cdot \boldsymbol{w}_{m',m''} + \boldsymbol{b}$, where $(\boldsymbol{w}; \boldsymbol{b})$ denotes shared weights and biases of convolutional core, and the size of convolutional core is $C \times C$.

The ACT layer is used to increase the nonlinear expression ability of the model. Therefore, nonlinear computation is involved in this layer. The activation function involved in the VGG16 model [11] is the rectified linear unit [i.e., $\text{ReLU}(\boldsymbol{y}) = \max(\boldsymbol{x}, 0)$].

The POOL layer is responsible for feature selection, i.e., a downsampling region is replaced by a single value. The maximum pooling (MAX-POOL) used in the VGG16 model takes the maximum value as the output $(\boldsymbol{x}_\delta, \delta = \arg\max\{\boldsymbol{x}_j, j = 1, \ldots, P \times P\})$.

For the FC layer, the goal is to map features to labeled samples, i.e., making the final classification decision. Each neuron in this layer is connected to all neurons in the anterior layer, and each link contains weight and bias parameters. Similar to the CONV layer, the core calculation is also a pure linear combination, i.e., $\boldsymbol{y}_j = \Sigma_k^F \boldsymbol{w}_{j,k} \cdot \boldsymbol{x}_k + \boldsymbol{b}$, where F is the number of neurons in the former layer.

C. *Additive Secret Sharing*

Secret sharing [29] is defined as follows. The trusted party divides a secret into N shares, which are kept by N participants, respectively. No less than $k (k \leq N)$ participants can reconstruct the complete secret. Somewhat differently, there are only two participants in the ASS [30], i.e., $N = k = 2$. The purpose of ASS is no longer limited to key management, but also can be used to protect all secrets contained in the integer domain.

In this article, we consider the array as minimum computing unit, and all elements are in the integral domain $\mathbb{Z}_n$, where n represents the size of $\mathbb{Z}_n$. The two edge servers $\mathcal{S}_1$ and $\mathcal{S}_2$ execute all proposed protocols independently or cooperatively. Here, we introduce some basic secure protocols for linear calculation, mainly secure addition/subtraction (SecAdd/SecSub) and secure scalar multiplication (SecSMul) protocols, leveraging ASS. The input array $(\boldsymbol{u}, \boldsymbol{v}) \in \mathbb{Z}_n$ of the SecAdd/SecSub protocol is split into $(\boldsymbol{u}_1, \boldsymbol{v}_1)$ and $(\boldsymbol{u}_2, \boldsymbol{v}_2)$, and sent to $\mathcal{S}_1$ and $\mathcal{S}_2$, respectively. Independently, $\mathcal{S}_1$ calculates $\boldsymbol{f}_1 = \boldsymbol{u}_1 \pm \boldsymbol{v}_1$ and $\mathcal{S}_2$ calculates $\boldsymbol{f}_2 = \boldsymbol{u}_2 \pm \boldsymbol{v}_2$. The simple addition of $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ gives the same result as the original addition/subtraction, because $\boldsymbol{f}_1 + \boldsymbol{f}_2 = \boldsymbol{u} \pm \boldsymbol{v}$ is always true. Similar to the SecAdd/SecSub protocol, the input array $(\boldsymbol{p}, \boldsymbol{q}) \in \mathbb{Z}_n$ of the SecSMul protocol is divided into $(\boldsymbol{p}_1, \boldsymbol{q})$ and $(\boldsymbol{p}_2, \boldsymbol{q})$, and $\mathcal{S}_1$ and $\mathcal{S}_2$ calculate $\boldsymbol{f}_1 = \boldsymbol{q} \cdot \boldsymbol{p}_1$ and $\boldsymbol{f}_2 = \boldsymbol{q} \cdot \boldsymbol{p}_2$, respectively. Obviously, $\boldsymbol{f}_1 + \boldsymbol{f}_2 = \boldsymbol{p} \cdot \boldsymbol{q}$.

III. System Architecture and Security Model

A. *System Architecture*

In the proposed framework, there are three major components: 1) CAVs (sending vehicles $\mathcal{V}_1$ and receiving

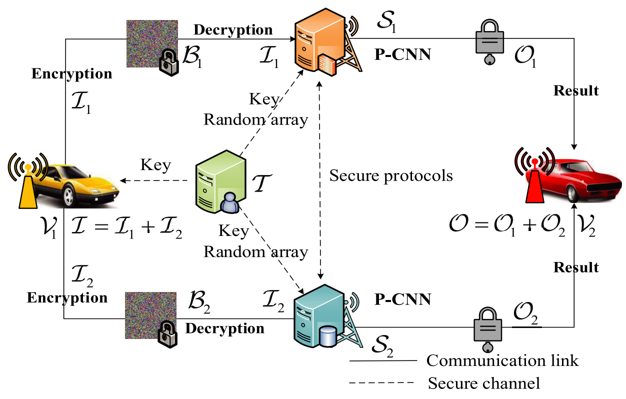

Fig. 2. System architecture of the privacy-preserving object classification for CAVs.

vehicles $\mathcal{V}_2$); 2) two edge servers ($\mathcal{S}_i, i = 1, 2$); and 3) a semitrusted lightweight server ($\mathcal{T}$), as shown in Fig. 2.

1) *$\mathcal{T}$:* $\mathcal{T}$ is responsible for generating the key for image encryption and transmitting it to the sending vehicle $\mathcal{V}_1$ and $\mathcal{S}_i$ through the secure channel. Moreover, $\mathcal{T}$ provides the random array for $\mathcal{S}_i$ to process the image/feature array.
2) *$\mathcal{V}_1$:* $\mathcal{V}_1$ perceives its surrounding environment, using HD sensors, and generates image data. Each image $\mathcal{I}$ is random split into two image shares $\mathcal{I}_i$ (i.e., $\mathcal{I} = \mathcal{I}_1 + \mathcal{I}_2$). Then, $\mathcal{V}_1$ encrypts them as encrypted images $\mathcal{B}_i$, and sends them to $\mathcal{S}_i$, respectively, so that $\mathcal{B}_i$ are not directly exposed to the communication link.
3) *$\mathcal{S}_i$:* Before image processing, $\mathcal{S}_i$ first needs to perform a round of decryption calculations to recover $\mathcal{I}_i$. Then, $\mathcal{S}_i$ executes the P-CNN model by using a serial secure computing protocols and obtains two classification result $\mathcal{O}_i$, respectively.
4) *$\mathcal{V}_2$:* After receiving $\mathcal{O}_i$, $\mathcal{V}_2$ only needs to perform simple addition calculation to recover the original classification results from the encrypted feature (i.e., $\mathcal{O} = \mathcal{O}_1 + \mathcal{O}_2$).

B. Security Model

In this article, we discuss the privacy-preserving inference process over encrypted image. The parameters of the neural network model are private to $\mathcal{S}_i$ and cannot be known by $\mathcal{V}_1$; The raw image, features extracted by $\mathcal{S}_i$ and final classification result are private to $\mathcal{V}_1$ and cannot be known by $\mathcal{S}_i$.

Similar to many existing literatures [23], [24], [31]–[33], we adopt a semihonest security model. $\mathcal{T}$ and $\mathcal{S}_i$ are considered as honest-but-curious (HBC) entities. They strictly follow the implementation protocol and are interested in the secret information owned by other entities. At the same time, we assume that $\mathcal{T}$ and $\mathcal{S}_i$ are independent and do not collude with each other [23], [24]. In practice, this assumption often manifests itself as $\mathcal{S}_1$ and $\mathcal{S}_2$ belonging to different or even competitive companies, such as Huawei's TaiShan@Edge and Microsoft Azure IoT Edge, as well as $\mathcal{T}$ can serve by some neutral agency. Since it is only responsible for generating/distributing encryption key and random array by secure channel in offline, $\mathcal{T}$ does not participate in the online calculation. This means that $\mathcal{T}$ does not receive/possess any meaningful image data, and does not affect framework security. Moreover, we also assume that the effective vehicles $\mathcal{V}_i$ sharing data are always reliable and the communication link do not be dropped.

In the semihonest model, there are many potential probabilistic polynomial time (PPT) adversaries $\mathcal{A}$ aimed at obtaining the image privacy captured by $\mathcal{V}_i$ using the onboard equipment. $\mathcal{A}$ has the following capabilities and constraints: 1) $\mathcal{A}$ can corrupt the sending vehicles and receiving vehicles to obtain the stored encrypted data, yet not obtain encryption key; 2) $\mathcal{A}$ can eavesdrop all communication links to obtain transmitted information, including image data sent by sending vehicle to edge servers and classification results sent by edge servers to receiving vehicle; 3) $\mathcal{A}$ can only corrupts and obtain the information of at most one of two edge servers; and 4) $\mathcal{A}$ cannot eavesdrop the secure channel between $\mathcal{T}$, $\mathcal{S}_1$, and $\mathcal{S}_2$, and cannot corrupt $\mathcal{T}$. The success of $\mathcal{A}$ attacking means that the complete original image or feature is obtained. This is actually quite difficult, as detailed security analysis is illustrated in Section V-C.

IV. Privacy-Preserving Object Classification for CAV

In this section, we describe the entire operation processes of the proposed privacy-preserving object classification framework in detail.

A. Image Reencryption

For the resisting simultaneous channel attack, this article introduces additional image encryption operation while randomly splitting the image in the image upload stage. Even if adversaries $\mathcal{A}$ are able to capture both upload links at the same time, the original image $\mathcal{I}$ cannot be recovered without knowing the encryption key. In fact, the process of image reencryption consists of two stage: 1) image split and 2) image confusion (i.e., real image encryption). First, $\mathcal{V}_1$ random splits each frame of its image into two images shares $\mathcal{I}_i$ leveraging ASS technologies. Then, $\mathcal{V}_1$ reencrypts $\mathcal{I}_i$ into two encrypted images $\mathcal{B}_i$. In order to not significantly increase the image upload cost, the image encryption scheme based on the chaos theory is designed to overcome this issue. Below, we illustrate the process of key generation and encryption of this scheme.

1) Key Generation: Combining logistic map and lorenz map, we construct a new efficient 3-D discrete chaotic map, as shown in formula (2), for generating encryption keys. The inputs tuple $(\boldsymbol{\gamma}_1(0), \boldsymbol{\gamma}_2(0), \boldsymbol{\gamma}_3(0))$ are within in the interval $(0, 1)$, $(-5, 5)$, and $(-60, 60)$, respectively. In order to make the constructed chaotic map in the chaotic state, we set its control parameter tuple $(\varepsilon_1, \varepsilon_2, \varepsilon_3)$ as $(3.7, 0.5, 0.3)$. After 4000 iterations, we can clearly observe that the constructed chaotic mapping sequence is almost disordered, and it is difficult to infer a trace of objective law, as shown in Fig. 3

$$\begin{cases} \boldsymbol{\gamma}_1(k+1) = \varepsilon_1 \boldsymbol{\gamma}_1(k) - {\boldsymbol{\gamma}_1}^2(k) \\ \boldsymbol{\gamma}_2(k+1) = \varepsilon_2 \boldsymbol{\gamma}_2(k) \boldsymbol{\gamma}_3(k) - \varepsilon_3 \boldsymbol{\gamma}_3(k) \\ \boldsymbol{\gamma}_3(k+1) = \boldsymbol{\gamma}_1(k) + \boldsymbol{\gamma}_2(k). \end{cases} \tag{2}$$

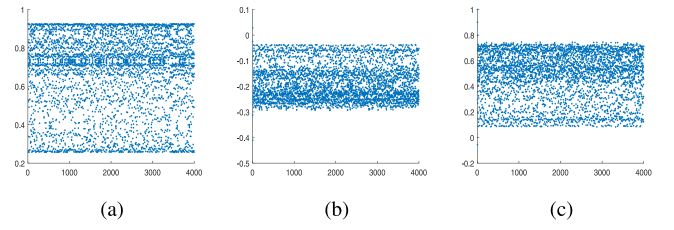

Fig. 3. Trajectory of our 3-D chaotic map after 4000 iterations. (a) $\boldsymbol{\gamma}_1$ sequence. (b) $\boldsymbol{\gamma}_2$ sequence. (c) $\boldsymbol{\gamma}_3$ sequence.

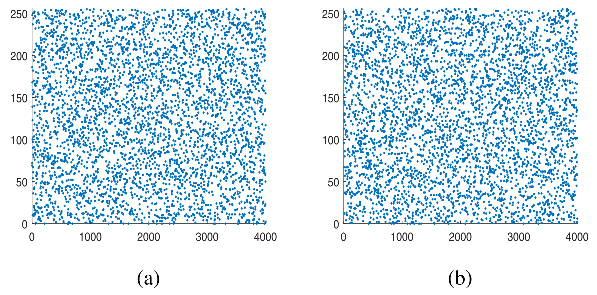

Fig. 4. Distribution of 4000 randomly selected from extended chaotic sequences. (a) $\boldsymbol{\delta}_1$ sequence. (b) $\boldsymbol{\delta}_2$ sequence.

Assuming the image size is (τ_1, τ_2, τ_3), we iterate the chaotic map in formula (2) for $\tau_1\tau_2\tau_3$ times, using the input $(\boldsymbol{\gamma}_1(0), \boldsymbol{\gamma}_2(0), \boldsymbol{\gamma}_3(0)) = (0.5, 0.5, 0.5)$. As a result, three chaotic sequences $\boldsymbol{\gamma}_1$, $\boldsymbol{\gamma}_2$, and $\boldsymbol{\gamma}_3$ with length $\tau_1\tau_2\tau_3$ are generated. Theoretically, the resulting chaotic sequence is aperiodic, but in practice, this assumption is very harsh. Therefore, we use a kind of sinusoidal function to deal with the above sequences, and then obtain a longer period of chaotic sequence $\boldsymbol{\varphi}_1 = \sin^2(\boldsymbol{\gamma}_1)$ and $\boldsymbol{\varphi}_2 = \sin^2(\boldsymbol{\gamma}_2 + \boldsymbol{\gamma}_3)/2$ in the interval [0, 1]. In order to obfuscate image pixel values, these sequence values are obviously too small and it is necessary to further generalize them to the entire pixel space [0, 255]. The transform $\boldsymbol{\delta}_1 = (\text{round}(10^{16}\boldsymbol{\varphi}_1) \mod 256$ and $\boldsymbol{\delta}_2 = (\text{round}(10^{16}\boldsymbol{\varphi}_2)) \mod 256$ are performed, where 10^{16} is the extended period. Without loss of generality, we randomly selected 4000 sequence values, and it can be seen from Fig. 4 that values of $\boldsymbol{\delta}_1$ and $\boldsymbol{\delta}_2$ sequences are approximately randomly distributed in pixel interval [0, 255]. Therefore, we choose these two sequences $\boldsymbol{\delta}_1$ and $\boldsymbol{\delta}_2$ as the initial encryption keys.

2) Encryption: Similar to the generated key, the images $\mathcal{I}_i(i = 1, 2)$ sent to $\mathcal{S}_1$ and $\mathcal{S}_2$ are stretched into stream sequence of length $\tau_1 \cdot \tau_2 \cdot \tau_3$. In order to make the encrypted image more sensitive, that is, to increase the dependency between ciphertext and plaintext-key pairing, we performed two rounds of encryption calculation sequentially, as illustrated in

$$\begin{cases} \mathcal{P}_i(1) = \mathcal{I}_i(1) \oplus \boldsymbol{\delta}_1(1) \\ \mathcal{P}_i(k) = ((\mathcal{I}_i(k) + \boldsymbol{\delta}_1(k-1)) \mod 256) \oplus \\ \qquad ((\boldsymbol{\delta}_1(k) + \mathcal{P}_i(k-1)) \mod 256) \end{cases} \tag{3}$$

$$\begin{cases} \mathcal{B}_i(1) = \mathcal{P}_i(1) \\ \mathcal{B}_i(k) = (\mathcal{P}_i(k)) \oplus ((\mathcal{B}_i(k-1) + \boldsymbol{\delta}_2(k) \mod 256) \\ \qquad \oplus\ \boldsymbol{\delta}_2(k-1)). \end{cases} \tag{4}$$

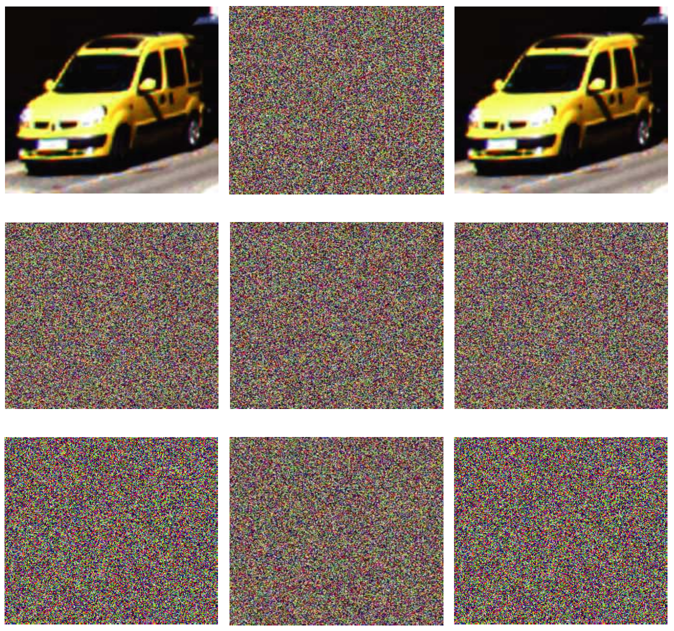

Fig. 5. Visual effect of encrypting and decrypting original image and two randomly segmented images. The top row belongs to $\mathcal{I}$, the middle row belongs to $\mathcal{I}_1$, and the bottom row belongs to $\mathcal{I}_2$. (a) Original image. (b) Encrypted image. (c) Recovered image.

$\mathcal{P}_i(i = 1, 2)$ are encrypted sequence after the first round of encryption, while $\mathcal{B}_i$ are encrypted sequence after the second round of encryption. Modular and XOR operations are performed on unsigned integers and, importantly, are reversible. As shown in Fig. 5, the original image $\mathcal{I}$ is randomly split into $\mathcal{I}_i$.

It is obvious that only image $\mathcal{I}_i$ cannot reveal any one pixel value, however, $\mathcal{I}$ can be recovered if both $\mathcal{I}_1$ and $\mathcal{I}_2$ are obtained at the same time. After two rounds of calculation, the images $\mathcal{I}_i$ are encrypted as $\mathcal{B}_i$. Even if the adversary master $\mathcal{B}_1$ and $\mathcal{B}_2$, it is unrealistic to decrypt and obtain $\mathcal{I}$ without the correct key.

B. Image Processing With P-CNN

After receiving encrypted images $\mathcal{B}_i$, $\mathcal{S}_i$ first needs to perform the reverse operation of encryption to recover the two image shares $\mathcal{I}_i$. Specifically, the first round of decryption is the reverse operation of the second round of encryption, and the second round of decryption is the reverse operation of the first round of encryption. The decryption key δ_1 and δ_2 are provided by $\mathcal{T}$ (see in Fig. 2), and the decryption is calculated as illustrated in formulas (5) and (6). As shown in Fig. 5(c), $\mathcal{S}_i$ can recover $\mathcal{I}_i$

$$\begin{cases} \mathcal{P}_i(k) = (\mathcal{B}_i(k)) \oplus ((\mathcal{B}_i(k-1) + \boldsymbol{\delta}_2(k) \mod 256) \\ \qquad \oplus \boldsymbol{\delta}_2(k-1)) \\ \mathcal{P}_i(1) = \mathcal{B}_i(1) \end{cases} \tag{5}$$

$$\begin{cases} \mathcal{I}_i(k) = (\mathcal{P}_i(k) \oplus (\mathcal{P}_i(k) + \boldsymbol{\delta}_1(k-1) \mod 256) \\ \qquad -\ \boldsymbol{\delta}_1(k-1)) \mod 256 \\ \mathcal{I}_i(1) = \mathcal{P}_i(1) \oplus \boldsymbol{\delta}_1(1). \end{cases} \tag{6}$$

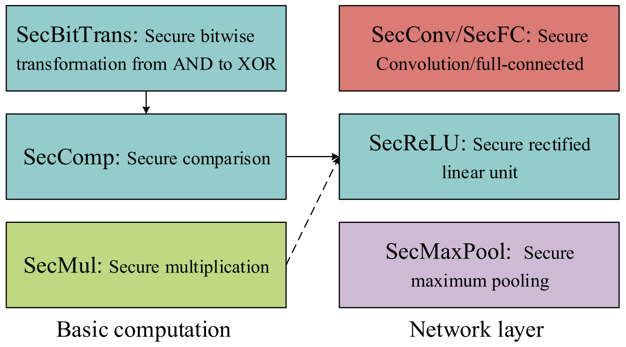
SecBitTrans: Secure bitwise transformation from AND to XOR
SecConv/SecFC: Secure Convolution/full-connected
SecComp: Secure comparison
SecReLU: Secure rectified linear unit
SecMul: Secure multiplication
SecMaxPool: Secure maximum pooling
Basic computation
Network layer

Fig. 6. Dependency relationship of protocols.

Afterward, $\mathcal{S}_i$ collaboratively executes P-CNN over $\mathcal{I}_i$ to extract ciphertext features, and finally obtain two classification shares $\mathcal{O}_i$. We mainly introduce the implementation process of each network layers in CNN. In view of the application scope of ASS, for each fixed-point number u, we multiply it with 10^κ, and transform it into $\overline{u}$ in integer domains $\mathbb{Z}_n$ through $\lfloor u \cdot 10^\kappa \rfloor$. In the absence of special instructions, the overline is omitted as follows.

Next, we mainly discuss the secure implementation methods of CONV layer, ReLU layer, MAX-POOL layer, and FC layer in P-CNN. In order to facilitate the understanding, the dependency relationship of protocols proposed in this article is shown in Fig. 6. Secure bitwise transformation from AND to XOR (SecBitTrans), secure comparison (SecComp), and secure multiplication (SecMul) [24] protocols is the component of the secure rectified linear unit (SecReLU) protocol.

1) Secure CONV and FC Layer: Since the CONV layer involves only linear operations, i.e., matrix dot product, the idea of ASS can be used directly. The 2-D convolution operation is shown in formula (7), where $\boldsymbol{x}$ represents feature matrix of size (d_1, d_2), $\boldsymbol{w}$ represents convolution kernel weight of size (e_1, e_2), and b represents convolution kernel bias. As the convolution kernel slides across $\boldsymbol{x}$ (from left to right and from top to bottom), $\boldsymbol{x}$ expands to $\bar{\boldsymbol{x}}$, and the convolution operation can be viewed as the dot product of the submatrix in $\bar{\boldsymbol{x}}$ with $\boldsymbol{w}$, finally returning convolution result $\boldsymbol{y}$ of size $(d_1 - e_1 + 1, d_2 - e_2 + 1)$. In P-CNN, $\boldsymbol{x}$ is split into two random shares $\boldsymbol{x}_i$ (i.e., $\bar{\boldsymbol{x}}_i$). Adopting the SecSMul protocol, $\mathcal{S}_i$ individually computes $\boldsymbol{y}_i = SecAdd(SecSMul(\bar{\boldsymbol{x}}_i, \boldsymbol{w}), b)$. Obviously, $\boldsymbol{y} = \boldsymbol{y}_1 + \boldsymbol{y}_2$ holds in any case

$$\begin{aligned}&\mathrm{Conv}\left(\boldsymbol{x} = \begin{bmatrix} x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 \\ x_7 & x_8 & x_9 \end{bmatrix}, \left(\boldsymbol{w} = \begin{bmatrix} w_1 & w_2 \\ w_3 & w_4 \end{bmatrix}, b\right)\right)\\ &= \left(\bar{\boldsymbol{x}} = \left[\left[\begin{bmatrix} x_1 & x_2 \\ x_4 & x_5 \end{bmatrix}\begin{bmatrix} x_2 & x_3 \\ x_5 & x_6 \end{bmatrix}\right]\left[\begin{bmatrix} x_4 & x_5 \\ x_7 & x_8 \end{bmatrix}\begin{bmatrix} x_5 & x_6 \\ x_8 & x_9 \end{bmatrix}\right]\right]\right) \odot \boldsymbol{w} + b\\ &= \begin{bmatrix} x_1w_1 + \cdots + x_5w_4 & x_2w_1 + \cdots + x_6w_4 \\ x_4w_1 + \cdots + x_8w_4 & x_5w_1 + \cdots + x_9w_4 \end{bmatrix} = \boldsymbol{y}. \quad (7)\end{aligned}$$

Similar to the CONV layer, the FC layer contains dot product operation. Given public connected weight and feature vector share, $\mathcal{S}_i$ can locally compute dot product result share by invoking the SecSMul protocol.

Protocol 1: Secure Bitwise Transformation From AND to XOR (SecBitTrans)

```
Input: $\mathcal{S}_i (i = 1, 2)$ has $\boldsymbol{u}_i \in \mathbb{Z}_n$.
Output: $\mathcal{S}_i$ outputs $\boldsymbol{f}_i$.
1 $\mathcal{T}$ randomly generates $\boldsymbol{\mu}_i \in \mathbb{Z}_n$, computes $\boldsymbol{\theta} \leftarrow \boldsymbol{\mu}_1 \otimes \boldsymbol{\mu}_2$,
  and randomly splits $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_1 \oplus \boldsymbol{\theta}_2$;
2 $\mathcal{T}$ sends $\boldsymbol{\mu}_i$ and $\boldsymbol{\theta}_i$ to $\mathcal{S}_i$;
3 $\mathcal{S}_i$ computes $\boldsymbol{\eta}_i \leftarrow \boldsymbol{u}_i \oplus \boldsymbol{\mu}_i$, and sends $\boldsymbol{\eta}_i$ to $\mathcal{S}_{3-i}$;
4 $\mathcal{S}_1$ computes $\boldsymbol{f}_1 \leftarrow \boldsymbol{\theta}_1 \oplus (\boldsymbol{\mu}_1 \otimes \boldsymbol{\eta}_2)$, $\mathcal{S}_2$ computes
  $\boldsymbol{f}_2 \leftarrow \boldsymbol{\theta}_2 \oplus (\boldsymbol{\mu}_2 \otimes \boldsymbol{\eta}_1) \oplus (\boldsymbol{\eta}_1 \otimes \boldsymbol{\eta}_2)$;
5 $\mathcal{S}_i$ returns $\boldsymbol{f}_i$.
```

Protocol 2: SecComp

```
Input: $\mathcal{S}_i$ has $\boldsymbol{u}_i \in \mathbb{Z}_n$.
Output: $\mathcal{S}_i$ outputs $\boldsymbol{f}_i$.
1 for $j = 0, \cdots, l-1$ do
2     $\mathcal{S}_i$ jointly compute $\boldsymbol{\chi}_i^{(j)} \leftarrow SecBitTrans(\boldsymbol{u}_1^{(j)}, \boldsymbol{u}_2^{(j)})$;
3 $\mathcal{S}_i$ locally computes $\boldsymbol{f}_i^{(0)} \leftarrow \boldsymbol{u}_i^{(0)}$ and $\boldsymbol{c}_i^{(0)} \leftarrow \boldsymbol{\chi}_i^{(0)}$;
4 for $j = 1, \cdots, l-1$ do
5     $\mathcal{S}_i$ jointly compute $\boldsymbol{\alpha}_i^{(j)} \leftarrow SecBitTrans(\boldsymbol{u}_1^{(j)}, \boldsymbol{c}_2^{(j-1)})$
      and $\boldsymbol{\beta}_i^{(j)} \leftarrow SecBitTrans(\boldsymbol{c}_1^{(j-1)}, \boldsymbol{u}_2^{(j)})$;
6     $\mathcal{S}_i$ locally computes
      $\boldsymbol{f}_i^{(j)} \leftarrow (\boldsymbol{u}_i^{(j)} \otimes \boldsymbol{c}_i^{(j-1)}) \oplus \boldsymbol{\chi}_i^{(j)} \oplus \boldsymbol{\alpha}_i^{(j)} \oplus \boldsymbol{\beta}_i^{(j)}$;
7 $\mathcal{S}_i$ returns $\boldsymbol{f}_i^{(l-1)}$.
```

2) Secure ReLU Layer: The ReLU function in CNN is used to selectively activate features; in other word, the goal is to suppress features less than zero, i.e., computing $\max(\boldsymbol{x}, 0)$. The problem is how to design SComp so that $\mathcal{S}_i$ can cooperatively compute the relationship between $\boldsymbol{x}$ and 0 without revealing the real features $\boldsymbol{x}$. In fact, this can be reduced to a positive–negative decision problem. To get most significant bit (MSB) sharing, we simply need to convert the addition sharing to exclusive-or (XOR) sharing. Using bool circuit, we can achieve bitwise addition operation with carry. Additionally, we specially design the SecBitTrans protocol based on ASS technology to provide carry for the above bitwise addition, aiming to convert the AND sharing to XOR sharing for single bit.

The construction of SecBitTrans protocol is shown in Protocol 1. Given input array shares $\boldsymbol{u}_i (i = 1, 2)$, participating in the calculation in the form of complement, the goal of the SecBitTrans protocol is to get $\boldsymbol{f}_i$, satisfying $\boldsymbol{f}_1 \oplus \boldsymbol{f}_2 = \boldsymbol{u}_1 \otimes \boldsymbol{u}_2$. Adopting the idea of Beaver triple [34], $\mathcal{T}$ establishes the relationship between $\boldsymbol{\mu}_i$ and $\boldsymbol{\theta}_i$, such that $\boldsymbol{\theta}_1 \oplus \boldsymbol{\theta}_2 = \boldsymbol{\mu}_1 \otimes \boldsymbol{\mu}_2$ (line 1). Among them, $\oplus$ represents XOR operation, and $\otimes$ represents AND operation. Note that these random numbers are selected from group $\mathbb{Z}_n$, where the bit length of n is typically 16, 32, or 64 bit, depending on the security requirement of the protocol. Then, $\mathcal{S}_i$ hides $\boldsymbol{u}_i$ in $\boldsymbol{\eta}_i$ before sending to $\mathcal{S}_{3-i}$; Finally, $\mathcal{S}_i$ converts AND sharing $\boldsymbol{u}_i$ into XOR sharing $\boldsymbol{f}_i$.

The detailed construction is shown in Protocol 2. Given input array shares $\boldsymbol{u}_i (i = 1, 2)$, participating in l-bit complement form $\boldsymbol{u}_i^{(l-1,\ldots,0)}$, SecComp protocol is objective to

Protocol 3: SecReLU

```
Input: $\mathcal{S}_i$ has input feature $\boldsymbol{x}_i \in \mathbb{Z}_n$.
Output: $\mathcal{S}_i$ outputs activation feature $\boldsymbol{y}_i$.
1 $\mathcal{S}_i$ jointly compute $(\boldsymbol{\zeta}_1, \boldsymbol{\zeta}_2) \leftarrow SecComp(\boldsymbol{x}_1, \boldsymbol{x}_2)$;
2 $\mathcal{S}_i$ jointly compute $(\boldsymbol{s}_1, \boldsymbol{s}_2) \leftarrow SecMul(\boldsymbol{\zeta}_1, 0, \boldsymbol{\zeta}_2, 1)$;
3 $\mathcal{S}_1$ locally computes $\boldsymbol{t}_i \leftarrow \boldsymbol{\zeta}_i + \boldsymbol{s}_i$;
4 $\mathcal{S}_i$ jointly compute $(\boldsymbol{y}_1, \boldsymbol{y}_2) \leftarrow SecMul(\boldsymbol{x}_1, 1 - \boldsymbol{t}_1, \boldsymbol{x}_2, -\boldsymbol{t}_2)$;
5 $\mathcal{S}_i$ returns $\boldsymbol{y}_i$.
```

Protocol 4: SecMaxPool

```
Input: $\mathcal{S}_i$ has input feature $\boldsymbol{x}_i \in \mathbb{Z}_n$.
Output: $\mathcal{S}_i$ outputs max-pooling feature $\boldsymbol{y}_i$.
1 foreach pooling region $R$ do
2     $\mathcal{S}_i$ initialize $\mathfrak{b}_1, \mathfrak{b}_2 \leftarrow 0$;
3     for $j, k$ in $(0, 0) \rightarrow (1, 1)$ do
4         $\mathcal{S}_i$ computes $\lambda_i \leftarrow \boldsymbol{x}_i^R(\mathfrak{b}_1, \mathfrak{b}_2) - \boldsymbol{x}_i^R(j, k)$, and
          sends $\lambda_1$ to $\mathcal{S}_{3-i}$;
5         $\mathcal{S}_i$ locally computes $\lambda \leftarrow \lambda_1 + \lambda_2$;
6         if $\lambda < 0$ then
7             $\mathcal{S}_i$ assign $\mathfrak{b}_1 \leftarrow j$ and $\mathfrak{b}_2 \leftarrow k$;
8     $\mathcal{S}_i$ computes $\boldsymbol{y}_i^R \leftarrow \boldsymbol{x}_i^R(\mathfrak{b}_1, \mathfrak{b}_2)$;
9 $\mathcal{S}_i$ returns $\boldsymbol{y}_i$.
```

output $\boldsymbol{f}_i$, satisfying $\boldsymbol{f}_1 \oplus \boldsymbol{f}_2 = \boldsymbol{u}_1 + \boldsymbol{u}_2$. First, $\mathcal{S}_i$ jointly computes the carry result sharing $\boldsymbol{\chi}_i^{(l-1,\ldots,0)}$ without carry (lines 1 and 2). For the least significant bit (LSB), $\boldsymbol{f}_i^{(0)}$ and $\boldsymbol{c}_i^{(0)}$ are the addition result sharing and carry result sharing, respectively (line 3). Where $j = 1, \ldots, l-1$, it satisfies $\boldsymbol{c}_1^{(j)} \oplus \boldsymbol{c}_2^{(j)} \leftarrow (\boldsymbol{u}_1^{(j)} \otimes \boldsymbol{u}_2^{(j)}) \oplus ((\boldsymbol{u}_1^{(j)} \oplus \boldsymbol{u}_2^{(j)}) \otimes (\boldsymbol{c}_1^{(j-1)} \oplus \boldsymbol{c}_2^{(j-1)}))$ and $\boldsymbol{f}_1^{(j)} \oplus \boldsymbol{f}_2^{(j)} \leftarrow (\boldsymbol{u}_1^{(j)} \oplus \boldsymbol{u}_2^{(j)}) \oplus (\boldsymbol{c}_1^{(j-1)} \oplus \boldsymbol{c}_2^{(j-1)})$; $\mathcal{S}_i$ can obtain $\boldsymbol{f}_i^{(l-1,\ldots,0)}$ by invoking the SecBitTrans protocol (lines 4–6). According to the MSB shares $\boldsymbol{f}_i^{l-1}$, we can infer the sign of input. Obviously, if $\boldsymbol{f}_1^{l-1} \oplus \boldsymbol{f}_2^{l-1} = 0$, then $\boldsymbol{u}_1 + \boldsymbol{u}_2 \geq 0$; otherwise, $\boldsymbol{u}_1 + \boldsymbol{u}_2 < 0$.

Therefore, we design the SecReLU protocol that embed the SComp protocol to realize the same effect as normal ReLU function. As illustrated in Protocol 3, given the input feature $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ (i.e., $\boldsymbol{x} = \boldsymbol{x}_1 + \boldsymbol{x}_2$), $\mathcal{S}_i$ cooperatively obtains the XOR shares $\boldsymbol{\zeta}_i$ of MSB by applying the SecBitExtra protocol (line 1). However, passing the MSB shares directly can reveal the distribution of activation features. In order to protect distribution privacy, the XOR sharing $\boldsymbol{\zeta}_i$ of MSB is usually converted to additive sharing $\boldsymbol{t}_i$ (lines 2 and 3), subjected to $\boldsymbol{t}_1 + \boldsymbol{t}_2 = \boldsymbol{\zeta}_1 + \boldsymbol{\zeta}_2 + \boldsymbol{\zeta}_1 \cdot \boldsymbol{\zeta}_2 = \boldsymbol{\zeta}_1 \oplus \boldsymbol{\zeta}_2$. Among them, the SecMul [24] protocol is based on Beaver triples [34]. Using the SecMul protocol again, $\mathcal{S}_i$ can obtain ReLU shares $\boldsymbol{y}_i$, satisfying $\boldsymbol{y}_1 + \boldsymbol{y}_2 = (\boldsymbol{x}_1 + \boldsymbol{x}_2) \cdot (1 - (\boldsymbol{t}_1 + \boldsymbol{t}_2))$. If $\boldsymbol{t}_1 + \boldsymbol{t}_2 = 0$, then $\boldsymbol{y}_1 + \boldsymbol{y}_2 = \boldsymbol{x}_1 + \boldsymbol{x}_2$; otherwise, $\boldsymbol{y}_1 + \boldsymbol{y}_2 = 0$.

3) Secure MAX-POOL Layer: The goal of the MAX-POOL layer is to calculate $\max\{\boldsymbol{x}(j,k), j, k = 0, 1\}$. To avoid disclosing feature privacy to $\mathcal{S}_i$, we designed a secure maximum pooling protocol (SecMaxPool) adopting the idea of ASS technology, as illustrated in Protocol 4. Given the input feature $\boldsymbol{x}_i$,

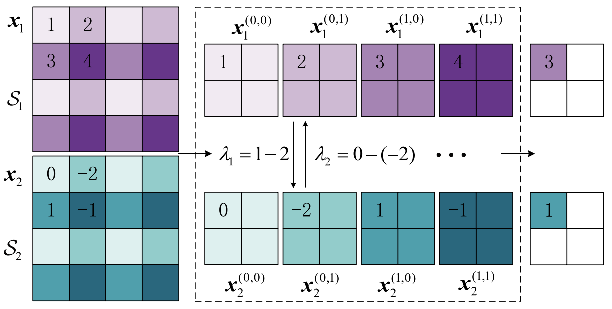
$\boldsymbol{x}_1$
1
2
3
4
$\mathcal{S}_1$
$\boldsymbol{x}_1^{(0,0)}$
$\boldsymbol{x}_1^{(0,1)}$
$\boldsymbol{x}_1^{(1,0)}$
$\boldsymbol{x}_1^{(1,1)}$
3
$\lambda_1 = 1 - 2$
$\lambda_2 = 0 - (-2)$
$\boldsymbol{x}_2$
0
-2
1
-1
$\mathcal{S}_2$
1
$\boldsymbol{x}_2^{(0,0)}$
$\boldsymbol{x}_2^{(0,1)}$
$\boldsymbol{x}_2^{(1,0)}$
$\boldsymbol{x}_2^{(1,1)}$

Fig. 7. Simple instance of Fast-MaxPool protocol.

Protocol 5: Fast-SecMaxPool

```
Input: $\mathcal{S}_i$ has four-dimensional feature $\boldsymbol{x}_i \in \mathbb{Z}_n$.
Output: $\mathcal{S}_i$ outputs max-pooling feature $\boldsymbol{y}_i$.
1 $\mathcal{S}_i$ performs transpose $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i.transpose(2, 3, 0, 1)$;
2 $\mathcal{S}_i$ blocks $\boldsymbol{x}_i$ into $(\boldsymbol{x}_i^{(0,0)}, \boldsymbol{x}_i^{(0,1)}, \boldsymbol{x}_i^{(1,0)}, \boldsymbol{x}_i^{(1,1)})$ according to
  the odd-even index;
3 $\mathcal{S}_i$ jointly compute
  $(\boldsymbol{y}_1, \boldsymbol{y}_2) \leftarrow where(((\boldsymbol{x}_1^{(0,0)} - \boldsymbol{x}_1^{(0,1)}) + (\boldsymbol{x}_2^{(0,0)} - \boldsymbol{x}_2^{(0,1)})) <$
  $0, ((\boldsymbol{x}_1^{(0,0)} - \boldsymbol{x}_1^{(0,1)}), (\boldsymbol{x}_2^{(0,0)} - \boldsymbol{x}_2^{(0,1)})))$;
4 $\mathcal{S}_i$ jointly execute as step 3 over $\boldsymbol{x}_i^{(1,0)}$ and $\boldsymbol{x}_i^{(1,1)}$;
5 $\mathcal{S}_i$ performs transpose $\boldsymbol{y}_i \leftarrow \boldsymbol{y}_i.transpose(2, 3, 0, 1)$;
6 $\mathcal{S}_i$ returns $\boldsymbol{y}_i$.
```

$\mathcal{S}_i$ jointly calculates the maximum value of all pooling regions sequentially. Within the particular 2×2 pooling region R, the 2-D index of the maximum position is initialized to $(\mathfrak{b}_1, \mathfrak{b}_2)$. $\mathcal{S}_1$ and $\mathcal{S}_2$ are not allowed to directly exchange their own feature values. Euphemistically, $\mathcal{S}_i$ calculates the differences shares λ_i between position $(\mathfrak{b}_1, \mathfrak{b}_2)$ and (j, k) (line 4). By transferring λ_i, $\mathcal{S}_i$ do not disclose the specific feature information, but can easily determine the relationship between the values in these two positions. After constantly updating the location index $(\mathfrak{b}_1, \mathfrak{b}_2)$ of the maximum value, $\mathcal{S}_i$ can get the maximum value shares $\boldsymbol{y}_i^R$ of the pooling region R (lines 6–8). After traversing all pooling regions, $\mathcal{S}_i$ can securely obtain the max-pooling feature share $\boldsymbol{y}_i$.

Optimization: Unfortunately, the input is generally a multidimensional array, and the SecMaxPool protocol does not effectively avoid the additional overhead associated with multitier loops. In order to take full advantage of the array as a unit of calculating, we design a fast SecMaxPool (Fast-MaxPool) protocol. As illustrated in Protocol 5, $\mathcal{S}_i$ first splits the input feature map $\boldsymbol{x}_i$ into four submaps $\max\{\boldsymbol{x}_i^{j,k}, j, k = 0, 1\}$ according to the even and odd superscripts (j, k) (line 2), and then calculates the difference between the corresponding positions by array subtraction. Afterward, $\mathcal{S}_i$ can obtain the comparison results between corresponding positions indirectly (lines 3 and 4). Especially, the additional transpose operation is to avoid that the split steps disrupt the original pooling order.

For better understanding, an instance of the Fast-SecMaxPool protocol is as shown in Fig. 7. It should be emphasized that $\mathcal{S}_1$ and $\mathcal{S}_2$ only know the difference result (λ_1, λ_2), but not $\{\boldsymbol{x}^{(j,k)}, j, k = 0, 1\}$. $\mathcal{S}_i$ can judge $(\boldsymbol{x}_1^{(0,0)} +$

$\boldsymbol{x}_2^{(0,0)}) > (\boldsymbol{x}_1^{(0,1)} - \boldsymbol{x}_2^{(0,1)})$ according to $\lambda = \lambda_1 + \lambda_2$. Specifically, such comparisons need to be performed three times.

C. Object Classification Results

Upon receiving classification shares $\mathcal{O}_i$ from the edge servers, vehicle $\mathcal{V}_2$ combines $\mathcal{O}_i$ to decrypt the information provided by the two P-CNN models on edge servers. According to the idea of ASS, simply adding $\mathcal{O}_i$ yields the same classification result $\mathcal{O}$ as what the original CNN will produce by processing the original image $\mathcal{I}$.

D. Feasibility for Multiparty Computation

To better understand the P-CNN framework, only the two-party setting (i.e., $\mathcal{S}_1$ and $\mathcal{S}_2$) is discussed above. However, in the CAVs system, two parties are not enough when one of the edge servers is offline or vehicle (CAV) is out of that coverage.

Thus, in this section, we discuss the executing P-CNN in multiparty setting (i.e., $\mathcal{S}_i, i \in \{1, 2, 3\}$). For instance of three-party setting, each private image $\mathcal{I}$ is randomly split three shares $\mathcal{I}_i$ ($\mathcal{I} = \sum_i \mathcal{I}_i$). Adopting $(3, 2)$ secret sharing, each edge server $\mathcal{S}_i$ has two shares of three shares $(\mathcal{I}_i, \mathcal{I}_j), j = (i+1) \mod 3$. Suppose $\mathcal{S}_3$ has some communication problem, and $\mathcal{S}_1$ and $\mathcal{S}_2$ can cooperatively perform P-CNN and recover $\mathcal{I} = \mathcal{I}_1' + \mathcal{I}_2'$ [i.e., (2, 2) secret sharing], where $\mathcal{S}_1$ calculates $\mathcal{I}_1' = \mathcal{I}_1 + \mathcal{I}_2$ and $\mathcal{S}_2$ calculates $\mathcal{I}_2' = \mathcal{I}_3$. This method can overcome huge communication overhead caused by real-time collaboration between multiple servers and can also effectively solve the soft handoff between CAV and edge servers in the CAVs system. Obviously, this method can also be applied to n-party ($n \geq 3$) setting. At this point, we adopt $(n, 2)$ secret sharing to randomly split image, while any two servers can cooperatively perform P-CNN, and $n - 2$ edge servers can be tolerated to lost connection. In fact, CAV only needs to select the two closest edge servers to establish a connection for privacy-preserving object classification task.

V. Theoretical Analysis

A. Complexity Analysis

First, we analyze the computational complexity of key generation, image encryption, and decryption. The key is a stream sequence composed of several chaotic sequence values. In order to encrypt the image with size $\tau_1\tau_2\tau_3$, the computational complexity of key generation depends on the image size, i.e., $O(\tau_1\tau_2\tau_3)$. While the iterative process of encryption and decryption only contains some modular and exclusive-or operations, so the computational complexity is also $O(\tau_1\tau_2\tau_3)$. Assume that the bit width of the input is l. Since the length of the key is one-to-one corresponding to the size of the image, $\mathcal{T}$ only needs one round of offline communication and bears $2\tau_1\tau_2\tau_3 l$ overhead for key transmission.

Next, we analyze the computational and communication complexity of secure protocols proposed in this article. Since $\mathcal{T}$ can complete the computation offline, we can ignore the its overhead. Assume that the input size for each secure protocol is n. For secure noninteractive protocols (i.e., SecAdd/SecSub, SecSMul), $\mathcal{S}_i$ can locally compute without interaction communication. $\mathcal{S}_i$ only needs to perform some same-dimensional linear computation, so the computational complexity of these protocols is only $O(n)$. As illustrated in the Tables I and II, the computational complexity of the SecComp protocol is the same as that of Huang *et al.* [24], which is related to bit width l. On the other hand, the communication complexity of the SecComp protocol is better than Huang, because of the reduced transmission message size per communication in the SecBitTrans protocol.

TABLE I
Comparisons of Computational Complexity

Protocol	VGG16 [11]	Huang [24]	P-CNN
SecBitTrans	-	-	$O(n)$
SecComp	-	$O(nl)$	$O(nl)$
SecReLU	$O(n)$	$O(nl)$	$O(nl)$
Fast-SecMaxPool	$O(3n/4)$	$O(3nl/4)$	$O(3n/4)$

TABLE II
Comparisons of Communication Complexity

Protocol	Huang [24]		P-CNN	
	Rounds	Communication	Rounds	Communication
SecBitTrans	-	-	1	nl
SecComp	$l+1$	$n(10l-4)$	l	$n(3l-2)$
SecReLU	$l+3$	$n(13l-4)$	$l+2$	$n(6l-2)$
Fast-SecMaxPool	$3n(l+3)/4$	$3n(13l-4)/4$	$3n/4$	$3nl/4$

Moreover, we analyze the complexity of each layer in P-CNN. The CONV layer and FC layer only contain addition and scalar multiplication, and $\mathcal{S}_i$ does not need to interactive with each other using SecAdd and SecSMul protocol. Compared with the VGG16 network [11], $\mathcal{S}_i$ does not need to perform additional computational operations, and its computational complexity is $O(e_1e_2(d_1-e_1+1)(d_2-e_2+1))$. The ReLU layer invokes the SecComp protocol to get MSB of input, as well as adopting the SecMul protocol [24] to convert XOR sharing of MSB to additive sharing type. The computational complexity of the SecReLU protocol is $O(ln)$. Except for the SecComp protocol, $\mathcal{S}_i$ needs two rounds to perform the SecMul protocol, costing $3nl$ overhead in the ReLU layer. In the MAX-POOL layer, $\mathcal{S}_i$ needs to perform three comparisons, so the communication round is three times than the ReLU layer. Because the SecMaxPool and Fast-SecMaxPool protocols in P-CNN use swap difference instead of the SecComp protocol, the communication overhead is lower than that of work [24].

B. Correctness Analysis

Due to the decomposability of linear computation, SecAdd/SecSub and SecSMul protocols are obviously correct. For the SecBitTrans protocol, we can disassemble the calculation process as follows. Since that $\boldsymbol{\theta}_1 \oplus \boldsymbol{\theta}_2 = \boldsymbol{\mu}_1 \otimes \boldsymbol{\mu}_2$, $\boldsymbol{\eta}_1 = \boldsymbol{u}_1 \oplus \boldsymbol{\mu}_1$, $\boldsymbol{\eta}_2 = \boldsymbol{u}_2 \oplus \boldsymbol{\mu}_2$, $\boldsymbol{f}_1 \oplus \boldsymbol{f}_2 = (\boldsymbol{\mu}_1 \oplus \boldsymbol{\eta}_1) \otimes (\boldsymbol{\mu}_2 \oplus \boldsymbol{\eta}_2) = \boldsymbol{u}_1 \otimes \boldsymbol{u}_2$ can be inferred. In the SecComp protocol, for $j = 0$, $\boldsymbol{f}^{(0)} = \boldsymbol{u}^{(0)}$; for $j = 1, \ldots, l-1$, $\boldsymbol{f}_1^{(j)} \oplus \boldsymbol{f}_2^{(j)} = (\boldsymbol{u}_1^{(j)} \oplus \boldsymbol{u}_2^{(j)}) \oplus (\boldsymbol{c}_1^{(j-1)} \oplus \boldsymbol{c}_2^{(j-1)})$. Therefore, $\boldsymbol{f}_1^{(l-1,\ldots,0)} \oplus \boldsymbol{f}_2^{(l-1,\ldots,0)} = \boldsymbol{u}_1 + \boldsymbol{u}_2$. Among this, $\boldsymbol{f}^{l-1} = \boldsymbol{f}_1^{l-1} \oplus \boldsymbol{f}_2^{l-1}$ is the MSB of input. Note that negative numbers will be calculated in the corresponding complement form. Therefore, SecBitTrans and SecComp protocols are correct.

In addition, we employ these secure protocols to realize normal calculation involved in the VGG16 network. The correct analysis is as follows. After calling the noninteractive protocols, the CONV and FC layers can execute correctly. For the ReLU layer, the SecComp protocol has proved to be correct. According to the XOR sharing $\boldsymbol{\zeta}_i$ of MSB, we can obtain the additive sharing $\boldsymbol{t}_i$ of MSB, satisfying $\boldsymbol{t}_1+\boldsymbol{t}_2 = \boldsymbol{\zeta}_1+\boldsymbol{\zeta}_2+\boldsymbol{\zeta}_1\cdot\boldsymbol{\zeta}_2$. The ReLU result $\boldsymbol{y} = \boldsymbol{x}\cdot(1-\boldsymbol{t})$. If $\boldsymbol{x} \geq 0$, then $\boldsymbol{t} = 0$, $1-\boldsymbol{t} = 1$, and $\boldsymbol{y} = \boldsymbol{x}$; otherwise, ($\boldsymbol{x} < 0$), then $\boldsymbol{t} = 1$, $1-\boldsymbol{t} = 0$, and $\boldsymbol{y} = 0$. For the MAX-POOL layer, the core operation is the three comparisons in SecMaxPool or Fast-SecMaxPool protocol. $\lambda = (\boldsymbol{x}_1(\mathfrak{b}_1, \mathfrak{b}_2) - \boldsymbol{x}_1(j,k)) + (\boldsymbol{x}_2(\mathfrak{b}_1, \mathfrak{b}_2) - \boldsymbol{x}_2(j,k)) = \boldsymbol{x}(\mathfrak{b}_1, \mathfrak{b}_2) - \boldsymbol{x}(j,k)$, if $\lambda < 0$, it means that $\boldsymbol{x}(\mathfrak{b}_1, \mathfrak{b}_2) < \boldsymbol{x}(j,k)$, and the index $(\mathfrak{b}_1, \mathfrak{b}_2)$ needs to update by (i,j). After the comparison, the index $(\mathfrak{b}_1, \mathfrak{b}_2)$ belongs to the maximum in pooling region R. Similarly, Fast-SecMaxPool opens up a new array space $(\boldsymbol{y}_1, \boldsymbol{y}_2)$ and iteratively updates the elements in the array until all values in the pooling region are compared. Specifically, $((\boldsymbol{x}_1^{(0,0)} - \boldsymbol{x}_1^{(0,1)}) + (\boldsymbol{x}_2^{(0,0)} - \boldsymbol{x}_2^{(0,1)})) = (\boldsymbol{x}^{(0,0)} - \boldsymbol{x}^{(0,1)})$; by comparison, the larger elements of the two arrays $\boldsymbol{x}^{(0,0)}$ and $\boldsymbol{x}^{(0,1)}$ are stored in $(\boldsymbol{y}_1, \boldsymbol{y}_2)$. Finally, $(\boldsymbol{y}_1, \boldsymbol{y}_2)$ stores the larger values for each of the four arrays $\{\boldsymbol{x}^{(0,0)}, \boldsymbol{x}^{(0,1)}, \boldsymbol{x}^{(1,0)}, \boldsymbol{x}^{(1,1)}\}$. Therefore, the processes of ReLU and MAX-POOL layers are correct.

Moreover, the calculations involved in the image split and object decryption are obvious correct, i.e., $\mathcal{I} = \mathcal{I}_1+\mathcal{I}_2$ and $\mathcal{O} = \mathcal{O}_1+\mathcal{O}_2$. Image encryption and decryption are a set of inverse operations, for the first pixel value of the image $\mathcal{I}_i$, $\mathcal{P}_i(1) = \mathcal{I}_i(1) \oplus \boldsymbol{\delta}_1(1)$, $B_i(1) = \mathcal{P}_i(1)$, and $\mathcal{I}_i(1) = \mathcal{P}_i(1) \oplus \boldsymbol{\delta}_1(1)$. For the remaining pixel values, the first round of encryption $\mathcal{P}_i(k) = ((\mathcal{I}_i(k)+\boldsymbol{\delta}_1(k-1)) \bmod 256)\oplus((\boldsymbol{\delta}_1(k)+\mathcal{P}_i(k-1)) \bmod 256)$ corresponds to the second round of decryption $\mathcal{I}_i(k) = (\mathcal{P}_i(k)\oplus(\mathcal{P}_i(k)+\boldsymbol{\delta}_1(k-1) \bmod 256)-\boldsymbol{\delta}_1(k-1)) \bmod 256$. While the second round of encryption $\mathcal{B}_i(k) = (\mathcal{P}_i(k)) \oplus ((\mathcal{B}_i(k-1) + \boldsymbol{\delta}_2(k) \bmod 256) \oplus \boldsymbol{\delta}_2(k-1))$ corresponds to the first round of decryption $\mathcal{P}_i(k) = (\mathcal{B}_i(k)) \oplus ((\mathcal{B}_i(k-1) + \boldsymbol{\delta}_2(k) \bmod 256) \oplus \boldsymbol{\delta}_2(k-1))$. Therefore, these encryption and decryption processes are correct.

C. Security Analysis

In the HBC model, two servers $\mathcal{T}$, $\mathcal{S}_1$, and $\mathcal{S}_2$ are not collusive. PPT adversary $\mathcal{A}$ can only corrupt and get the view (i.e., feature information) of at most one of $\mathcal{S}_1$ and $\mathcal{S}_2$. To better demonstrate security, we formalize a definition as follows.

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

Lemma 1: A protocol is perfectly simulatable if all its subprotocols are perfectly simulatable [35].

Lemma 2: If a random element a is uniformly distributed on $\mathbb{Z}_n$ and independent from any variable $b \in \mathbb{Z}_n$, then $a \pm b$ is also uniformly random and independent from b [36].

Since our protocols can be simulated in practice, we need to learn from above lemmas to assist the proof process.

Theorem 1: The SecAdd/SecSub and SecSMul protocols are secure in the HBC model.

Proof: For the SecAdd/SecSub protocol, the real view $\text{View}_{\text{real}}$ of $\mathcal{S}_1$ and $\mathcal{S}_2$ is $\{\boldsymbol{u}_1, \boldsymbol{v}_1, \boldsymbol{f}_1\}$ and $\{\boldsymbol{u}_2, \boldsymbol{v}_2, \boldsymbol{f}_2\}$. The simulator $\mathcal{M}$ can easily generate the corresponding simulation view View_{sim}, and $\text{View}_{\text{real}}$ are computationally indistinguishable from View_{sim} for $\mathcal{A}$. Similarly, $\text{View}_{\text{real}}$ of $\mathcal{S}_1$ and $\mathcal{S}_2$ in the SecSMul protocol are $\{\boldsymbol{p}_1, \boldsymbol{q}\}$ and $\{\boldsymbol{p}_2, \boldsymbol{q}\}$. The input $\boldsymbol{q}$ is a public value. Therefore, the SecAdd/SecSub and SecSMul protocols are proved to be secure in the HBC model. ■

Theorem 2: The SecBitTrans and SecComp protocols are secure in the HBC model.

Proof: For the SecBitTrans protocol, the inputs $\boldsymbol{u}_i (i = 1, 2)$ are uniformly random. From the perspective of $\mathcal{S}_i$, $\text{View}_{\text{real}}$ is $\{\boldsymbol{u}_i, \boldsymbol{\mu}_i, \boldsymbol{\theta}_i, \boldsymbol{\eta}_i, \boldsymbol{\eta}_{3-i}, \boldsymbol{f}_i\}$. Since $\{\boldsymbol{\mu}_i, \boldsymbol{\theta}_i\}$ are random number offline generated by $\mathcal{T}$, $\boldsymbol{\eta}_i$ are also uniformly random according to Lemma 1. $\mathcal{S}_i$ cannot infer the real inputs $\boldsymbol{u}$ from $\boldsymbol{\eta}_i + \boldsymbol{\eta}_i$, but obtain the output $\boldsymbol{f}_i$. As a consequence, the real view $\text{View}_{\text{real}}$ is simulatable for $\mathcal{M}$ and computationally indistinguishable from the simulated view View_{sim} for $\mathcal{A}$. Since the SecBitTrans protocol can be simulated, the SecComp protocol is simulatable according to Lemma 1. $\text{View}_{\text{real}}$ of $\mathcal{S}_i$ is $\{\boldsymbol{u}_i, \boldsymbol{\chi}_i, \boldsymbol{\alpha}_i, \boldsymbol{\beta}_i, \boldsymbol{f}_i\}$. Also, the SecBitTrans protocol has proved to be secure, and $\{\boldsymbol{\chi}_1, \boldsymbol{\alpha}_i, \boldsymbol{\beta}_1\}$ are uniformly random. Finally, $\mathcal{S}_i$ can be obtain the output $\boldsymbol{f}_i$, subjected to $\boldsymbol{f}_1 \oplus \boldsymbol{f}_2 = \boldsymbol{u}_1 + \boldsymbol{u}_2$. Obviously, the MSB $\boldsymbol{f}_i^{l-1}$ is also uniformly random. $\mathcal{M}$ can generate simulation view View_{sim} of $\mathcal{S}_i$, which is computationally indistinguishable from $\text{View}_{\text{real}}$. Thus, SecBitTrans and SecComp protocols are secure in the HBC model. ■

Theorem 3: The SecReLU, SecMaxPool, and Fast-SecMaxPool protocols are secure in the HBC model.

Proof: Since the security of the SecComp protocol has been proved in Theorem 1, the security of the SecMul protocol has been proved in work [24]. Clearly, $\{\boldsymbol{\zeta}_i, \boldsymbol{s}_i, \boldsymbol{t}_i, \boldsymbol{y}_i\}$ are uniformly random. Since SecComp and SecMul protocol can be simulated, the SecReLU protocol is simulatable by $\mathcal{M}$ according to Lemma 1. View_{sim} is computationally indistinguishable from $\text{View}_{\text{real}}$ of $\mathcal{S}_i$. For the SecMaxPool protocol, the inputs $\boldsymbol{x}_i$ are uniformly random. $\text{View}_{\text{real}}$ of $\mathcal{S}_i$ is $\{\boldsymbol{x}_i, \lambda_1, \lambda_2, \mathfrak{b}_1, \mathfrak{b}_2, \boldsymbol{y}_i\}$. Even if $\mathcal{S}_i$ gets the difference λ_{3-i} and λ (i.e., $x(\mathfrak{b}_1, \mathfrak{b}_2) - \boldsymbol{x}(j,k)$), it cannot recover $\boldsymbol{x}(j,k)$. As a result, for $\mathcal{A}$, $\text{View}_{\text{real}}$ and View_{sim}, which is generated by $\mathcal{M}$, are computationally indistinguishable. For the Fast-SecMaxPool protocol, the input is same as the SecMaxPool protocol. $\text{View}_{\text{real}}$ of $\mathcal{S}_i$ is $\{\boldsymbol{x}_i^{(0,0)}, \boldsymbol{x}_i^{(0,1)}, \boldsymbol{x}_i^{(1,0)}, \boldsymbol{x}_i^{(1,1)}, \boldsymbol{y}_i\}$. This means that $\mathcal{A}$ needs to corrupt both $\mathcal{S}_1$ and $\mathcal{S}_2$ to recover $\{\boldsymbol{x}^{(0,0)}, \boldsymbol{x}^{(0,1)}, \boldsymbol{x}^{(1,0)}, \boldsymbol{x}^{(1,1)}, \boldsymbol{y}\}$, however, $\mathcal{S}_1$ or $\mathcal{S}_2$ are assumed not to be collusive, so the $\mathcal{A}$ cannot obtain meaningful information. Thus, the SecReLU, SecMaxPool, and Fast-SecMaxPool protocols are secure in the HBC model. ■

Theorem 4: The image reencryption, image decryption, image process with P-CNN, and object decryption are secure in the HBC model.

Proof: In the process of image reencryption, a vehicle $\mathcal{V}_1$ randomly splits the image $\mathcal{I}$ into $\mathcal{I}_1$ and $\mathcal{I}_2$ and encrypts them into $\mathcal{B}_1$ and $\mathcal{B}_2$. Then, $\mathcal{V}_1$ sends the ciphertext image $\mathcal{B}_1$

Fig. 8. Instance of manual KITTI data sets.

and $\mathcal{B}_2$ to $\mathcal{S}_1$ and $\mathcal{S}_2$, respectively. If the adversary corrupts $\mathcal{S}_1$ or $\mathcal{S}_2$, obtaining $\mathcal{I}_1$ or $\mathcal{I}_2$ cannot restore the original image $\mathcal{I}$. However, if $\mathcal{A}$ is capable of corrupting the communication links between $\mathcal{V}_1$ and $\mathcal{S}_1$ as well as $\mathcal{V}_1$ and $\mathcal{S}_2$ at the same time, or even directly corrupting $\mathcal{V}_1$, then $\mathcal{A}$ can easily obtain the original image $\mathcal{I}$. This is a very practical security problem, and the additional encryption scheme described in Section IV-A is designed to avoid such a bad situation. Since $\mathcal{B}_1 + \mathcal{B}_2 \neq \mathcal{I}$, even if both $\mathcal{B}_1$ and $\mathcal{B}_2$ are obtained, $\mathcal{A}$ cannot obtain any meaningful information, depending on the security of the encryption scheme. The detailed security experiment is conducted in Section VI-A. In the process of object decryption, $\mathcal{S}_1$ and $\mathcal{S}_2$ send the extracting feature $\mathcal{O}_1$ and $\mathcal{O}_2$ to the receiver vehicle $\mathcal{V}_2$. $\mathcal{V}_2$ can secure obtain the classification result $\mathcal{O}$ by calculating $\mathcal{O}_1 + \mathcal{O}_2$, without leaking any image privacy. Additionally, P-CNN employs SecAdd, SecSub, and SecSMul protocols to perform CONV and FC operation, and employs SecReLU and SecMaxPool (or FastMaxPool) protocols to perform ReLU and MAX-POOL operations. The above protocols have been proved to be secure; thus, the image process with P-CNN is also secure. ■

VI. Experiment and Results

As our framework is designed for the CAVs system, we choose the real-world KITTI data set [37] to train and test our model. In order to match object classification task, we process the KITTI data set by cropping out manually the vehicles and pedestrians from each frames provided in the data set, as shown in Fig. 8. The customized data set contains 3000 training samples and 750 test samples, with the size $(224, 224, 3)$ of each sample. Similar to the previous privacy-preserving schemes, in this article, we train the network in plaintext and inference over encrypted image. All experiments are simulated on the personal computer of Ubuntu 14.04 with the following hardware configurations, Intel Xeon E5-10 GPU with 4-GB RAM, and 32-GB main RAM, and softwares, deep learning framework CAFFE, Interpreter Python3.5, and IDE PyCharm. Specifically, the package Numpy is used as a multidimensional container of numbers to execute our secure protocol in parallel.

Next, we first discuss the actual security of the image encryption (decryption) described in Section IV-A (Section IV-B) and analyze it in detail from the aspects of key sensitivity, pixel correlation, and differential attack. Then, we evaluate the proposed secure protocols in terms of computational cost and communication overhead, which directly affects the efficiency of P-CNN. In order to fully show the performance of our framework, we also monitor the performance of each network layer in real time and make a comprehensive comparison with the existing excellent literature. Each experiment result is represented by the mean value of repeated runs of ten times.

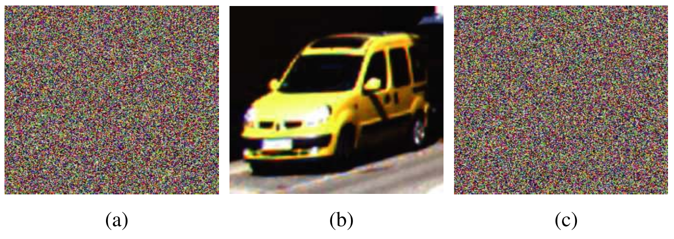
Fig. 9. Visual effect of original image and encrypted images. (a) Encrypted image encrypted with key$_{\text{real}}$. (b) Original image decrypted with key$_{\text{real}}$. (c) Recover image decrypted with key$_{\text{fict}}$.

A. Security of Image Encryption

1) Key Sensitivity Analysis: In our image encryption scheme, the encryption key is related to the value of $(\boldsymbol{\gamma}_1(0), \boldsymbol{\gamma}_2(0), \boldsymbol{\gamma}_3(0))$; here, we set it to $(0.5, 0.5, 0.5)$. In general, a secure encryption scheme relies heavily on key sensitivity, using two slightly different keys to decrypt an image, and the results are quite different. For encrypted images encrypted with key key$_{\text{real}} = (0.5, 0.5, 0.5)$, in order to fully reveal the sensitivity of the key, we decrypt them with the correct key key$_{\text{real}}$ and the fictitious key key$_{\text{fict}} = (0.5 + 10^{-8}, 0.5, 0.5)$, respectively. As shown in Fig. 9, the difference between the fictitious key key$_{\text{fict}}$ and the correct key key$_{\text{real}}$ is only 10^{-8}, but the correct original image cannot be recovered, and any characteristic information of image cannot be obtained.

In fact, adversary $\mathcal{A}$ can only decrypt the correct image when the fictitious key key$_{\text{fict}} = (\boldsymbol{\gamma}'_1(0), \boldsymbol{\gamma}'_2(0), \boldsymbol{\gamma}'_3(0))$ simultaneously satisfies $|\boldsymbol{\gamma}'_1(0) - \boldsymbol{\gamma}_1(0)| \le 10^{-8}$, $|\boldsymbol{\gamma}'_2(0) - \boldsymbol{\gamma}_2(0)| \le 10^1$, and $|\boldsymbol{\gamma}'_3(0) - \boldsymbol{\gamma}_3(0)| \le 10^2$. It is extremely difficult for $\mathcal{A}$ to construct a suitable set of keys over a large range of real numbers. If the unlimited space of the key is $[10^{-10}, 10^{10}]^3$, then the probability that $\mathcal{A}$ decrypts correctly is 10^{-35}, which is a very low probability. Obviously, our encryption scheme is sufficient to resist against the exhaustive attacks with existing computer capabilities.

2) Pixel Correlation Analysis: Pixel correlation is a important statistical characteristic measuring image encryption schemes. The strong correlation between image pixels threatens the security of image information. The lower the correlation, the higher the degree of pixel value confusion. As shown in Fig. 10, the adjacent pixel values in the original image $\mathcal{I}$ have a strong linear relationship. In fact, this means that the adjacent pixel values have a strong correlation. Given the pixel values at a certain location, it is reasonable to infer the adjacent pixel values. In contrast, when the image is encrypted, the correlation between the pixel values at any two locations is almost identical, which means that the above conjectures are no longer possible. The relationship between adjacent pixel values becomes sharp, and the characteristic information of the original image is well disturbed.

3) Differential Attack Analysis: Differential attack usually means that the potential adversary extracts some information about the key by changing the original image $\mathcal{I}$ slightly, or

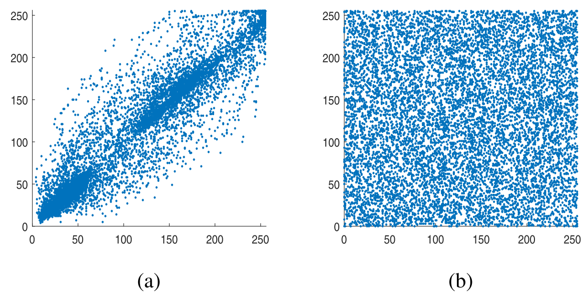

Fig. 10. Pixel correlation. (a) Pixel correlation of original image $\mathcal{I}$. (b) Pixel correlation of encrypted image $\mathcal{B}$.

TABLE III
COMPARISONS OF NPCR AND UACI

Scheme	Image	NPCR (%)	UACI (%)
Chen [38]	-	99.60	33.47
Hu [39]	-	99.60	33.38
P-CNN	$\mathcal{I}_1$	99.61	33.56
	$\mathcal{I}_2$	99.62	33.49
	$\mathcal{I}$	99.60	36.64

even only one pixel value, and then comparing it with the encrypted image $\mathcal{B}$. In order to make the difference attack infeasible, a minor modification in $\mathcal{I}$ should lead to the substantial difference in $\mathcal{B}$. Generally speaking, the number of pixel change rate (NPCR) and unit of average change intensity (UACI) are used to measure the impact of one pixel change in $\mathcal{I}$ on $\mathcal{B}$. The larger NPCR and NPCR are, the higher the degree of pixel confusion is. Suppose that $\mathcal{I}(k)$ and $\mathcal{B}(k)$ be the kth pixel of $\mathcal{I}$ and $\mathcal{B}$, and NPCR and UACI are defined in

$$\text{NPCR} = \frac{\sum_{k=1}^{(\tau_1,\tau_2,\tau_3)} \mathbb{I}(k)}{\tau_1 \times \tau_2 \times \tau_3} \times 100\% \tag{8}$$

$$\mathbb{I}(k) = \begin{cases} 0, & \text{if } \mathcal{I}(k) = \mathcal{B}(k) \\ 1, & \text{if } \mathcal{I}(k) \neq \mathcal{B}(k) \end{cases} \tag{9}$$

$$\text{UACI} = \frac{\sum_{k=1}^{(\tau_1,\tau_2,\tau_3)} |\mathcal{I}(k) - \mathcal{B}(k)|/255}{\tau_1 \times \tau_2 \times \tau_3} \times 100\%. \tag{10}$$

As shown in Table III, the NPCR of these schemes has little difference. In the other hand, compared with the image encryption method based on lookup tables [38] and that based on single Fibonacci chaotic mapping [39], P-CNN using the fusion of three chaotic sequences has a greater intensity of disrupting image pixels (i.e., UACI). After two rounds of encryption calculation, the UACI of P-CNN is as high as 33.49%. Obviously, P-CNN can resist differential attack better than these schemes [38] and [39].

B. Performance of Proposed Secure Protocols

These secure computing protocols are designed to securely implement P-CNN, and the main factors affecting their performance include: parallel batch size and bit-width l of inputs. From Fig. 11(a) and (b), the runtime of the SecBitTrans and SecComp protocols increases with the parallel batch size. Even if 10^5 inputs are processed in parallel, the running time of the SecBitTrans and SecComp protocols can be controlled within 4 and 16 ms, respectively. Compared with the SecCmp protocol proposed in [24], the efficiency of our SecComp protocol becomes more clear with the increase of batch size. This is because that we use two-tuples operator to realize the conversion from additive sharing to XOR sharing, rather than the Beaver triple operator, thus saving about 1/3 of the cost. SecReLU protocol can securely and correctly obtain the output of ReLU function by using SecComp and SecMul protocols. From Fig. 11(c), the runtime of the SecReLU protocol increases with the parallel batch size, slightly higher than the original ReLU function, but better than work [24]. Additionally, SecMaxPool protocol adopts the Naive method of swap difference instead of using SecComp protocol in MAX-POOL layer, reducing computational cost, as shown in Fig. 11(d). Furthermore, Fast-SecMaxPool protocol enhances the parallelization efficiency on the basis of SecMaxPool protocol, and achieves a slightly higher computational cost than the original MAX-POOL function.

From Fig. 11(e) and (f), the communication overhead of SecBitTrans and SecComp protocol increases with the bit width l of inputs. Compared with the SecCmp protocol proposed in [24], our SecComp protocol can reduce about 2/3 of the communication overhead. From Fig. 11(g) and (h), the communication overhead of SecReLU and SecMaxPool/Fast-SecMaxPool protocols increases with the bit width l of inputs. Importantly, these protocols proposed in this article are better than work [24] in the aspect of communication overhead.

Moreover, the computational error of SecComp protocol is negligible. Since the fractional portion of the fixed-point number is converted to the integer domain for computation, SecComp Protocol can accurately determine the relationship between the input and 0. Therefore, SecReLU protocol has no computational error using the SecComp protocol. Additionally, since no approximate calculation is involved in CONV, POOL, and FC layers, their error is always maintained at 0.

TABLE IV
COMPARISON OF RUNTIME FOR KEY GENERATION, IMAGE ENCRYPTION, AND DECRYPTION

Scheme	Key generation (s)	Encryption (s)	Decryption (s)
TFCM [40]	12.0	0.052	0.087
Hu [39]	-	1.13	-
P-CNN	0.168	0.686	0.655

C. Performance of Object Classification Framework

In the encryption phase, the sender vehicle $\mathcal{V}_1$ needs to randomly split the original image $\mathcal{I}$ into two images $\mathcal{I}_1$ and $\mathcal{I}_2$, and then encrypts them to obtain the ciphertext image $\mathcal{B}_1$ and $\mathcal{B}_2$. After receiving the ciphertext image, the two servers $\mathcal{S}_1$ and $\mathcal{S}_2$ decrypt them independently. As shown in Table IV, 0.686 s is required for $\mathcal{V}_1$ to encrypt one image, and 0.655 s is required for $\mathcal{S}_1$ and $\mathcal{S}_2$ to decrypt one image. Note that $\mathcal{S}_1$ and $\mathcal{S}_2$ are executed independently and in parallel, so it takes only time to decrypt one image. The comparison result shown that our encryption efficiency is higher than that of [39]. The first schemes [40] seem to be more efficient at encrypting and decrypting, but generating keys takes a significant amount of time, i.e., 12.0 s, which is not allowed in a dynamic communication environment. In contrast, only 0.168 s is required in

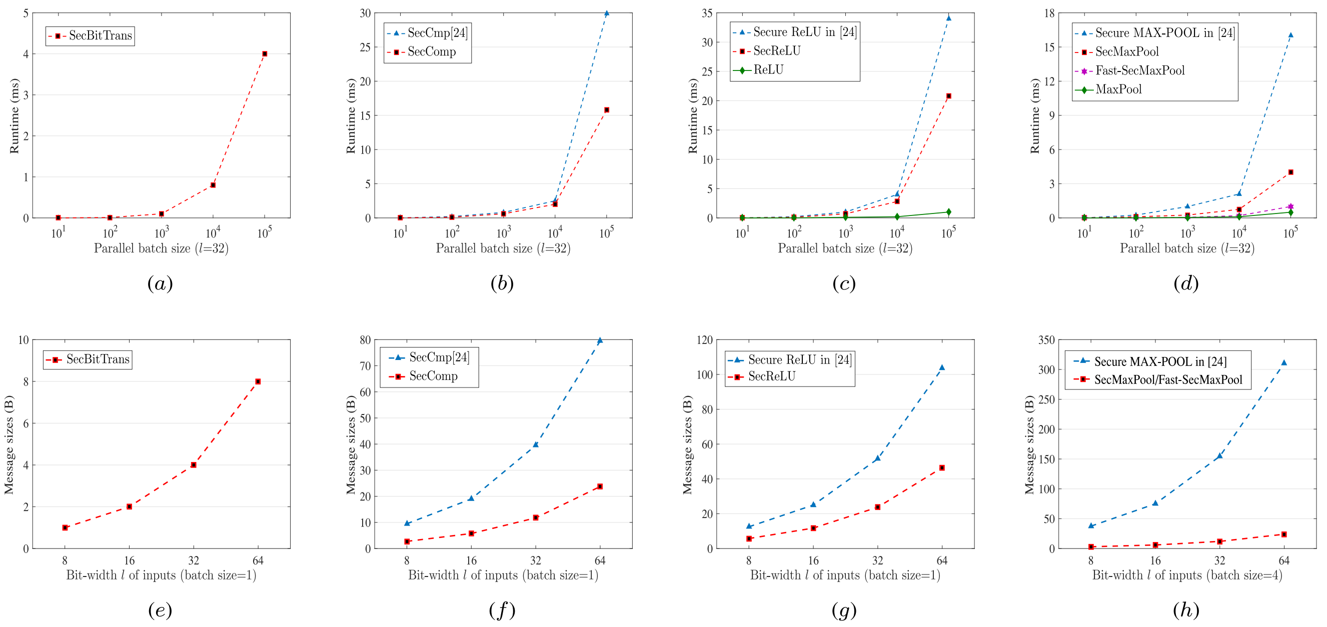

Fig. 11. Efficiency and effectiveness of the proposed secure computing protocols. (a) Runtime of SecBitTrans protocol varies from batch size. (b) Runtime of SecComp protocol varies from batch size. (c) Runtime of SecReLU protocol varies from batch size. (d) Runtime of Fast-SecMaxPool protocols varies from batch size. (e) Communication overhead of SecBitTrans protocol varies from bit width. (f) Communication overhead of SecComp protocol varies from bit width. (g) Communication overhead of SecReLU protocol varies from bit width. (h) Communication overhead of Fast-SecMaxPool protocol varies from bit-width.

TABLE V
COMPARISON OF THE COMMUNICATION OVERHEAD IN ENCRYPTION AND DECRYPTION PHASES FOR A BATCH OF 750 IMAGES FROM THE KITTI [37]

Scheme	Encryption		Decryption	
	Message size (MB)	Size per instance (MB)	Message size (KB)	Size per instance (KB)
CryptoNets [20]	4306.64	5.74	175.781	0.234
P-CNN	143.55	0.19	5.859	0.008

TABLE VI
COMPARISON OF THE RUNTIME FOR CONV AND FC

Scheme	Layer	Input	Output	Time (ms)
CryptoNets [20]	CONV	28 × 28 × 1	13 × 13 × 5	3000
CryptoDL [21]		28 × 28 × 1	24 × 24 × 20	1308
GAZELLE [22]		28 × 28 × 1	24 × 24 × 5	20.60
FALCON [41]		28 × 28 × 1	24 × 24 × 5	3.35
P-CNN		224 × 224 × 64	224 × 224 × 64	4.64
CryptoNets [20]	FC	100	10	160
CryptoDL [21]		800	10	3432
GAZELLE [22]		2048	1	24.20
FALCON [41]		2048	1	1.3
P-CNN		800	10	1.64

TABLE VII
COMPARISON OF THE RUNTIME FOR RELU AND MAX-POOL

Scheme	Layer	Feature size	Time (ms)
CryptoDL [21]	ReLU	24 × 24 × 20, 8 × 8 × 50, 500	976.3
Huang [24]			113.01
P-CNN			25.3
P-CNN		224 × 224 × 64	314.23
Huang [24]	POOL	24 × 24 × 20, 8 × 8 × 50	171.71
P-CNN			8.8
P-CNN		224 × 224 × 64	26.79

our encryption scheme, which means that the keys used for encryption can be updated at any time for greater security. Additionally, receiver vehicle $\mathcal{V}_2$ requires only the addition of the two outputs $\mathcal{O}_1$ and $\mathcal{O}_2$ in the decryption phase, consuming only 0.03 s.

On the other hand, the communication overhead in encryption and decryption phases are 143.55 and 5.859 MB, as shown in Table V. The images consist of 224 × 224 pixels, and each pixel is stored as 4 bytes. Using HE scheme, each pixel is encrypted as five polynomials, and each coefficient in the polynomial requires 24 bytes. Our communication overhead is far less than CryptoNets [20].

Moreover, we evaluate the computational efficiency of each layers in P-CNN, and compare with the previous work, as shown in Tables VI and VII. CryptoNets [20], CryptoDL [21], and Gazelle [22] mainly use HE to realize the calculation in the neural network, and FALCON [41] combines the fast fourier transform (FFT). Obviously, our runtime in CONV and FC layers is far less than CryptoNets [20], CryptoDL [21], and Gazelle [22]. Also, our scheme can deal with feature map of larger size (i.e., 224 × 224 × 64 in CONV layer and 800 in FC layer) than scheme FALCON [41]. In the same amount of time, although the work [24] also applies the idea of ASS, the computational complexity of comparison calculation involved in our SecReLU protocol is much less. For the MAX-POOL layer, the work [24] uses the proposed comparison function several times to determine the maximum value in the pooling region, while P-CNN directly finds the maximum value through the difference between the two pairs. Obviously, the cost of this calculation method is very small.

TABLE VIII
COMPARISON OF THE PERFORMANCE FOR DIFFERENT PRIVACY-PRESERVING SCHEME

Scheme	Runtime (s)	Message size (MB)
MiniONN [42]	9.32	657.5
FALCON [41]	0.84	92.5
GAZELLE [22]	0.81	70.0
Huang [24]	0.30	2.56
P-CNN	0.07	0.77

TABLE IX
COMPARISON OF THE PERFORMANCE FOR P-CNN AND ORIGINAL VGG16 MODEL

Network	Computational cost(s)	Message size (MB)	Accuracy
VGG16 [43]	2.22	-	96.93%
P-CNN	22.78	327.78	96.93%

Privacy-preserving network adopts the designed secure layer protocols to perform privacy computing over encrypted image. In order to facilitate comparison, we specially execute the smaller CNN network, consisting of two CONV layers, three ACT layers, two MAX-POOL layers, and two FC layers. The performance comparison results of different schemes are shown in Table VIII. Obviously, running the same network, P-CNN consumes less computing cost and communication overhead than the previous work. Furthermore, we utilize more complex VGG16 [43] model to implement P-CNN. As shown in Table IX, P-CNN can achieve the same classification accuracy as the original VGG16 model. The computational cost of P-CNN is 10.26× times than the VGG16 model, with an additional communication overhead of 327.78 MB. Synthetically, P-CNN has great advantages in computing and communication efficiency, and it is more realistic to execute in the edge server.

VII. RELATED WORK

Most of the existing image encryption schemes are based on symmetric encryption [38] or chaos theory [27], [28]. The symmetric encryption over image involves complicated byte or bit displacement, and the construction principle of S-box cannot be explained in detail. Generating keys and encryption usually leads to a large amount of overhead, which is not suitable for real-time environment such as CAV.

In contrast, the image encryption based on the chaos theory has higher encryption efficiency, and the generated ciphertext image has good randomicity in pixel value distribution. Run-He *et al.* [44] proposed a color image encryption algorithm based on the mixed diffusion mechanism and chaos, which extends the 2-D logical map to the 4-D chaotic map to replace the image pixel value. However, there is a certain degree of distortion in the decrypted recovered image. Kaur *et al.* [40] proposed a color image encryption (TFCM) using nondominated sorting genetic algorithm-based 5-D chaotic map, which decomposed the input color image into three channels, and obtained the final encrypted image through dual-tree complex wavelet transform. The experiments show that the TFCM scheme can resist against exhaustive and differential attacks, but the time to generate the key is slow. Hu *et al.* [39] proposed a color image encryption algorithm based on Fibonacci chaotic system, which scrambles the image pixel by using the chaotic sequence, and performs mutual exclusion-or operation on adjacent pixel values to further reduce the pixels correlation.

On the task of privacy-preserving image classification, the existing research work has made a breakthrough to some extent. CryptoNets [20] is the first privacy-preserving neural network model using leveled HE (LHE). However, it only supports simple secure linear operations, such as the activation layer using the square function, which is now rarely used in network training and prediction. Therefore, some polynomial functions [21] are designed to reproduce the commonly used activation functions, such as ReLU, Sigmoid, and Hyperbolic tangent (Tanh). SecureML [45] proposed the first privacy-preserving protocol for training multiple machine learning models between two noncollusive servers, which uses piecewise linear function to replace activation function. MiniONN [42] transformed a neural network model into an oblivious version, and adopts lattice-based AHE to generate multiplication triplets. Meanwhile, this scheme puts some computation offline, and greatly improves the calculation efficiency.

Gazelle [22] used a judicious combination of packed AHE (PAHE) and garbled circuit-based secure two-party computation, vectorizing the designed protocol, and further improving framework efficiency to some extent. Falcon [41] developed fast and secure protocols for convolutional and fully connected layers-based FFT, which is a special solution with higher computational efficiency than the secure protocol based on HE. Huang *et al.* [24] proposed a new solution based on addition secret sharing, and designed a SecComp function to force two noncolluding edge servers to perform the activation layer and pooling layer. The experimental results show that the computational cost and communication overhead of our solution are much lower than that of the models based on HE. Furthermore, Ma *et al.* [46], [47] implemented privacy-preserving speech recognition and face recognition using SComp protocols in work [24], and Liu *et al.* [48] proposed a secure object detection model with image and feature privacy protection. The optimization of the SComp protocol is meaningful for these models applying privacy-preserving tasks.

Based on the existing work, this article specifically designs a chaotic mapping-based image encryption scheme to avoid channel privacy leakage. In addition, we design a lightweight and secure computing protocol based on ASS technology, instead of using HE and garbled circuit with high computational cost.

VIII. CONCLUSION

Focusing on object classification problem, we proposed a lightweight, privacy-preserving cooperative object classification framework for CAV. The key innovation of the framework is the introduction of the P-CNN model, which processes encrypted images shared from vehicles and produces meaningful classification results when combined. The additional image obfuscation is to ensure the security of the image during storage and transmission. In P-CNN, the operations in the original CNN model (VGG16) are implemented by our secure protocols, based on the ASS technique. The experimental results demonstrate that P-CNN offers exactly the same classification results as the VGG16 model, without requiring vehicles to send original image data directly, which may contain privacy information. Moreover, the computational cost and communication overhead on two edge servers are acceptable.

In future work, we expect to design an adaptive addition cycle mechanism to further reduce the runtime involved in SecComp calculation. Additionally, we need to construct a secure detection paradigm combined with ASS, which can be applied to range positioning [49], 2-D-object detection [50], 3-D-object detection [51], etc. Potentially, we need to extend ASS to three or more parties for processing to achieve a higher level of security for CAVs.

References

[1] J. Van Brummelen, M. O'Brien, D. Gruyer, and H. Najjaran, "Autonomous vehicle perception: The technology of today and tomorrow," *Transp. Res. C, Emerg. Technol.*, vol. 89, pp. 384–406, Apr. 2018.

[2] L. Hu, Y. Qian, J. Chen, X. Shi, J. Zhang, and S. Mao, "Photo crowdsourcing based privacy-protected healthcare," *IEEE Trans. Sustain. Comput.*, vol. 4, no. 2, pp. 168–177, Apr.–Jun. 2019.

[3] Q. Yang, A. Lim, S. Li, J. Fang, and P. Agrawal, "ACAR: Adaptive connectivity aware routing for vehicular ad hoc networks in city scenarios," *Mobile Netw. Appl.*, vol. 15, no. 1, pp. 36–60, 2010.

[4] Q. Chen, S. Tang, Q. Yang, and S. Fu, "Cooper: Cooperative perception for connected autonomous vehicles based on 3D point clouds," in *Proc. 39th IEEE Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Dallas, TX, USA, 2019, pp. 514–524.

[5] Q. Chen, X. Ma, S. Tang, J. Guo, Q. Yang, and S. Fu, "F-Cooper: Feature based cooperative perception for autonomous vehicle edge computing system using 3D point," in *Proc. 4th ACM/IEEE Symp. Edge Comput. (SEC)*, 2019, pp. 88–100.

[6] Q. Liu, "An improved approach to exposing JPEG seam carving under recompression," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 7, pp. 1907–1918, Jul. 2019.

[7] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, "Communication-efficient federated learning and permissioned blockchain for digital twin edge networks," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2276–2288, Feb. 2021.

[8] J. Ni, X. Lin, and X. Shen, "Toward privacy-preserving valet parking in autonomous driving era," *IEEE Trans. Veh. Technol.*, vol. 68, no. 3, pp. 2893–2905, Mar. 2019.

[9] D. Glancy, "Privacy in autonomous vehicles," *Santa Clara Law Rev.*, vol. 52, no. 4, pp. 1171–1239, 2012.

[10] Q. Jiang, J. Ni, J. Ma, L. Yang, and X. Shen, "Integrated authentication and key agreement framework for vehicular cloud computing," *IEEE Netw.*, vol. 32, no. 3, pp. 28–35, May/Jun. 2018.

[11] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran, 2012, pp. 1097–1105.

[12] S. Liu, L. Liu, J. Tang, B. Yu, Y. Wang, and W. Shi, "Edge computing for autonomous driving: Opportunities and challenges," *Proc. IEEE*, vol. 107, no. 8, pp. 1697–1716, Aug. 2019.

[13] Y. Liu, S. Xie, and Y. Zhang, "Cooperative offloading and resource management for UAV-enabled mobile edge computing in power IoT system," *IEEE Trans. Veh. Technol.*, vol. 69, no. 10, pp. 12229–12239, Oct. 2020.

[14] J. Ni, K. Zhang, and A. V. Vasilakos, "Security and privacy for mobile edge caching: Challenges and solutions," *IEEE Wireless Commun.*, early access, Dec. 21, 2020, doi: 10.1109/MWC.001.2000329.

[15] 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.

[16] B. Yang, D. Wu, and R. Wang, "CUE: An intelligent edge computing framework," *IEEE Netw.*, vol. 33, no. 3, pp. 18–25, May/Jun. 2019.

[17] M. Chen, Y. Qian, J. Chen, K. Hwang, S. Mao, and L. Hu, "Privacy protection and intrusion avoidance for cloudlet-based medical data sharing," *IEEE Trans. Cloud Comput.*, vol. 8, no. 4, pp. 1274–1283, Oct./Dec. 2020.

[18] R. Lu, X. Lin, H. Zhu, P.-H. Ho, and X. Shen, "ECPP: Efficient conditional privacy preservation protocol for secure vehicular communications," in *Proc. IEEE INFOCOM 27th Conf. Comput. Commun.*, Phoenix, AZ, USA, 2008, pp. 1229–1237.

[19] B. Jin, D. Jiang, J. Xiong, L. Chen, and Q. Li, "D2D data privacy protection mechanism based on reliability and homomorphic encryption," *IEEE Access*, vol. 6, pp. 51140–51150, 2018.

[20] P. Xie, M. Bilenko, T. Finley, R. Gilad-Bachrach, K. Lauter, and M. Naehrig, "Crypto-nets: Neural networks over encrypted data," 2014. [Online]. Available: arXiv:1412.6181.

[21] E. Hesamifard, H. Takabi, and M. Ghasemi, "CryptoDL: Deep neural networks over encrypted data," 2017. [Online]. Available: arXiv:1711.05189.

[22] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, "GAZELLE: A low latency framework for secure neural network inference," in *Proc. 27th USENIX Security Symp. (USENIX Security)*, 2018, pp. 1651–1669.

[23] J. Xiong, R. Bi, M. Zhao, J. Guo, and Q. Yang, "Edge-assisted privacy-preserving raw data sharing framework for connected autonomous vehicles," *IEEE Wireless Commun.*, vol. 27, no. 3, pp. 24–30, Jun. 2020.

[24] 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.*, vol. 18, no. 3, pp. 1441–1455, May/Jun. 2021.

[25] C. Luo, J. Ji, Q. Wang, X. Chen, and P. Li, "Channel state information prediction for 5G wireless communications: A deep learning approach," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 1, pp. 227–236, Jan.–Mar. 2020.

[26] E. Ott, C. Grebogi, and J. A. Yorke, "Controlling chaos," *Phys. Rev. Lett.*, vol. 64, no. 11, p. 1196, 1990.

[27] N. K. Pareek, V. Patidar, and K. K. Sud, "Image encryption using chaotic logistic map," *Image Vis. Comput.*, vol. 24, no. 9, pp. 926–934, 2006.

[28] K. Mischaikow and M. Mrozek, "Chaos in the lorenz equations: A computer assisted proof," *Bull. Amer. Math. Soc.*, vol. 32, no. 1, pp. 66–72, 1995.

[29] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[30] 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.*, 2006, pp. 285–304.

[31] Z. Qin, J. Yan, K. Ren, C. W. Chen, and C. Wang, "Towards efficient privacy-preserving image feature extraction in cloud computing," in *Proc. 22nd ACM Int. Conf. Multimedia*, 2014, pp. 497–506.

[32] T. Araki, J. Furukawa, Y. Lindell, A. Nof, and K. Ohara, "High-throughput semi-honest secure three-party computation with an honest majority," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2016, pp. 805–817.

[33] J. Wang, S. Hu, Q. Wang, and Y. Ma, "Privacy-preserving outsourced feature extractions in the cloud: A survey," *IEEE Netw.*, vol. 31, no. 5, pp. 36–41, Sep./Oct. 2017.

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

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

[36] D. Bogdanov, M. Niitsoo, T. Toft, and J. Willemson, "High-performance secure multi-party computation for data mining applications," *Int. J. Inf. Security*, vol. 11, no. 6, pp. 403–418, 2012.

[37] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, 2013.

[38] J.-X. Chen, Z.-L. Zhu, C. Fu, L.-B. Zhang, and Y. Zhang, "An efficient image encryption scheme using lookup table-based confusion and diffusion," *Nonlinear Dyn.*, vol. 81, no. 3, pp. 1151–1166, 2015.

[39] X. Hu, L. Wei, W. Chen, Q. Chen, and Y. Guo, "Color image encryption algorithm based on dynamic chaos and matrix convolution," *IEEE Access*, vol. 8, pp. 12452–12466, 2020.

[40] M. Kaur, D. Singh, K. Sun, and U. Rawat, "Color image encryption using non-dominated sorting genetic algorithm with local chaotic search based 5D chaotic map," *Future Gener. Comput. Syst.*, vol. 107, pp. 333–350, Jun. 2020.

[41] S. Li *et al.*, "FALCON: A fourier transform based approach for fast and secure convolutional neural network predictions," 2018. [Online]. Available: arXiv:1811.08257.

[42] J. Liu, M. Juuti, Y. Lu, and N. Asokan, "Oblivious neural network predictions via MiniONN transformations," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2017, pp. 619–631.
[43] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015. [Online]. Available: arXiv:1409.1556.
[44] Q. Run-He, C. Yun, and F. Yu-Zhen, "Integrated confusion-diffusion mechanisms for chaos based image encryption," in *Proc. 4th Int. Congr. Image Signal Process.*, vol. 2. Shanghai, China, 2011, pp. 629–632.
[45] P. Mohassel and Y. Zhang, "SecureML: A system for scalable privacy-preserving machine learning," in *Proc. IEEE Symp. Security Privacy (SP)*, San Jose, CA, USA, 2017, pp. 19–38.
[46] Z. Ma, Y. Liu, X. Liu, J. Ma, and F. Li, "Privacy-preserving outsourced speech recognition for smart IoT devices," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8406–8420, Oct. 2019.
[47] 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.
[48] Y. Liu, Z. Ma, X. Liu, S. Ma, and K. Ren, "Privacy-preserving object detection for medical images with faster R-CNN," *IEEE Trans. Inf. Forensics Security*, early access, Oct. 10, 2019, doi: 10.1109/TIFS.2019.2946476.
[49] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran, 2015, pp. 91–99.
[50] J. Xiong, R. Bi, Q. Chen, and X. Liu, "Towards edge-collaborative, lightweight and secure region proposal network," *J. Commun.*, vol. 41, no. 10, pp. 188–201, 2020.
[51] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3D object detection network for autonomous driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Honolulu, HI, USA, 2017, pp. 1907–1915.

Jinbo Xiong (Member, IEEE) received the Ph.D. degree in computer system architecture from Xidian University, Xi'an, China, in 2013.

He was a Visiting Scholar with the Department of Computer Science and Engineering, University of North Texas, Denton, TX, USA. He is currently a Full Professor and the Ph.D. supervisor with the Fujian Provincial Key Laboratory of Network Security and Cryptology and the College of Computer and Cyber Security, Fujian Normal University, Fuzhou, China. He has published more than 100 publications, containing four ESI highly cited papers and two monographs and holds more than ten patents in these fields. His research interests include secure deep learning, data security, privacy protection, and Internet of Things.

Prof. Xiong serves as the TPC Chair of the 14th EAI Mobimedia and a PC member of numerous international conferences. He served or is serving as a Lead Guest Editor and/or a Guest Editor for several technical journals, such as *Security and Communication Networks*, *Wireless Communications and Mobile Computing*, *Concurrency and Computation: Practice and Experience*, and *Mobile Network and Applications*.

Renwan Bi received the M.S. degree in software engineering from the College of Mathematics and Informatics, Fujian Normal University, Fuzhou, China, in 2021, where he is currently pursuing the Ph.D. degree in cyberspace security with the College of Computer and Cyber Security.

His research interests include connected and autonomous vehicle, deep neural network, secure multiparty computation, and cryptography security.

Youliang Tian (Member, IEEE) received the Ph.D. degree in cryptography from Xidian University, Xi'an, China, in 2012.

He is a Full Professor and the Ph.D. supervisor with the State Key Laboratory of Public Big Data and the College of Computer Science and Technology, Guizhou University, Guiyang, China. He is a Young Changjiang Scholar of the Ministry of Education. He has authored over 100 publications and two books. His current research interests include algorithmic game theory, cryptography and security protocols, big data security and privacy protection, blockchain, and electronic currency.

Ximeng Liu (Senior Member, IEEE) received the B.Sc. degree in electronic engineering and the Ph.D. degree in cryptography from Xidian University, China, in 2010 and 2015, respectively.

He is currently a Full Professor and the Ph.D. supervisor with the College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China. He was a Research Fellow with the School of Information System, Singapore Management University, Singapore. He has published more than 250 papers on the topics of cloud security and big data security, including papers in IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, IEEE INTERNET OF THINGS JOURNAL, and so on. His research interests include cloud security, applied cryptography, and big data security.

Prof. Liu Awards the Minjiang Scholars Distinguished Professor, the Qishan Scholars in Fuzhou University, and the ACM SIGSAC China Rising Star Award in 2018.

Dapeng Wu (Senior Member, IEEE) received the M.S. degree in communication and information system from Chongqing University of Posts and Telecommunications, Chongqing, China, in 2006, and the Ph.D. degree in communication and information system from Beijing University of Posts and Telecommunications, Beijing, China, in 2009.

He is currently a Full Professor and the Ph.D. supervisor with the Chongqing University of Posts and Telecommunications. He has authored over 100 publications and two books. He is the inventor and co-inventor of 28 patents and patent applications. His current research interests include social computing, wireless networks, and big data.

Prof. Wu serves as the TPC Chair of the 10th Mobimedia and a PC member of numerous international conferences and workshops. He served or is serving as an Editor and/or a Guest Editor for several technical journals, such as *Digital Communications and Networks* (Elsevier) and *ACM/Springer Mobile Network and Applications*.