Accurate Image-Based Pedestrian Detection With Privacy Preservation

Haomiao Yang, *Member, IEEE*, Qixian Zhou, Jianbing Ni, *Member, IEEE*, Hongwei Li, *Senior Member, IEEE*, and Xuemin Shen, *Fellow, IEEE*

***Abstract*—In this paper, we propose an accurate pedestrian detection scheme with privacy preservation (PPPD) based on pedestrian images. By utilizing the vector homomorphic encryption (VHE), private linear-transforming matrices are ingeniously designed to enable arbitrary permutation operations for the encrypted vectors. In this way, the feature vector extraction of the histogram of oriented gradient (HOG) can be efficiently performed over the encrypted pedestrian images. Furthermore, due to the encrypted inner product calculations supported by VHE, an encrypted kernel matrix is constructed to generate multiple encrypted kernels (i.e., linear, polynomial, and Gaussian kernels). The pedestrian detection model based on the supported vector machine (SVM) can be securely trained over the encrypted kernels. With the proposed scheme, the extracted features of pedestrian images are not necessary to be returned to the image owner for decryption, such that the communication costs can be significantly reduced. In addition, the privacy of the whole process in pedestrian detection can also be guaranteed. Extensive experiments are conducted over multiple pedestrian datasets, and it is demonstrated that PPPD can achieve high accuracy of pedestrian detection with lower computation and communication overhead compared with the existing schemes.**

***Index Terms*—Pedestrian detection, vector homomorphic encryption, support vector machine, histogram of oriented gradient, privacy preservation.**

I. Introduction

TRAFFIC accidents involving pedestrians and other road users (e.g., bicyclists) are one of the leading causes of death and injury on roads [1]. To reduce these accidents, the pedestrian detection system is a critical component for the modern autonomous and semi-autonomous vehicles. It is capable of identifying the pedestrian while traversing through the terrain, such that the vehicles can take appropriate actions to avoid the collision in advance. As a result, road safety can be significantly improved. Pedestrian detection is not only applicable for autonomous vehicles [2], [3], [4], but also suitable to other emerging applications, such as video surveillance [5], image retrieval [6] and human-robot interaction [7].

Unfortunately, pedestrian detection in road images is pretty challenging to achieve. A typical approach is to leverage image feature extraction algorithms (e.g., the histogram of oriented gradient (HOG)) and machine learning algorithms (e.g., the support vector machine (SVM)) to identify the pedestrians from the videos and images captured by the cameras on vehicles [8]. Specifically, the HOG algorithm can maintain invariance of image geometric and optical deformation, and reflect the contour of the human body in image feature extraction. As HOG is insensitive to brightness and color change of an image, it is adaptive to various complex scenes in autonomous driving. The SVM algorithm has excellent classification performance due to the support of multiple kernel functions (linear, polynomial and Gaussian kernels), fitting high-dimensional HOG features. Therefore, by combining HOG and SVM, the features of the images can be properly extracted, and the pedestrian detection models can be trained based on the massive images.

To guarantee high accuracy in pedestrian detection, a large number of pedestrian images need to be analyzed, which brings heavy computation overhead to the vehicles. Therefore, it is challenging to implement all the steps of pedestrian detection (i.e., HOG feature extraction, SVM model training, and pedestrian identification) in resource-limited autonomous vehicles. Outsourcing heavy training tasks and pedestrian images to the cloud is an effective solution to offloading the computation overhead for vehicles [9]. With the aid of the cloud, the operations of pedestrian detection are classified into two categories: remote and local. The expensive computations of feature extracting and model training are carried out on the remote cloud; the lightweight detection is locally performed on the self-driving car. The cloud collects the pedestrian data in distinct situations to train or update the pedestrian detection model, and the autonomous vehicles make decisions based on the parameters of the trained pedestrian detection model in the cloud. The integration of remote training and local decision-making not only fully

Manuscript received May 16, 2020; revised September 17, 2020; accepted November 27, 2020. Date of publication December 8, 2020; date of current version January 22, 2021. This work was supported in part by the National Key R&D Program of China under Grants 2017YFB0802300, 2017YFB0802000, and 2017YFB0802003 and in part by the National Natural Science Foundation of China under Grant 62072081. The review of this article was coordinated by Prof. Yi Qian. *(Corresponding author: Jianbing Ni.)*

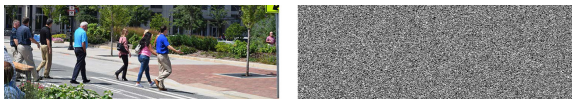
Haomiao Yang is with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China, and also with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: haomyang@uestc.edu.cn).

Qixian Zhou and Hongwei Li are with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China (e-mail: qxzhou1010@qq.com; hongweili@uestc.edu.cn).

Jianbing Ni is with the Department of Electrical and Computer Engineering, Queen's University, Kingston, ON K7L 3N6, Canada (e-mail: j25ni@uwaterloo.ca).

Xuemin Shen is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: xshen@bbcr.uwaterloo.ca).

Digital Object Identifier 10.1109/TVT.2020.3043203



(a) Original image (b) Encrypted image

Fig. 1. Image encryption. (a) Original image (b) Encrypted image.

makes use of the powerful computing and storage resources in the cloud, but also reduces the latency of decision-making [10].

However, the cloud suffers from a variety of cyberattacks, such as malware injection attacks, denial of service attacks, and side channel attacks. Also, the frequently occurred data exposure accidents seriously increase the risks of outsourced data leakage. These vulnerabilities result in huge concerns on the confidentiality of pedestrian data [11]. Moreover, it is possible for a curious adversary to extract private information about pedestrians from pedestrian images, such as location, background, or even companion. To prevent privacy leakage, encrypting pedestrian images prior to outsourcing alleviates privacy concerns. Nevertheless, after the images are encrypted, we cannot identify any knowledge from the encrypted images. As shown in Fig. 1, the traditional ciphers (e.g., AES [12]) would hide all the information in the clear images, which causes the difficulty of pedestrian detection over the encrypted images. To protect pedestrian privacy, Zhang *et al.* designed an anonymous camera based on the optical obscuration of the pedestrian's face [13]. Unfortunately, the anonymous camera still suffers from person re-identification attacks based on deep ranking [14]. Low-resolution pedestrian images can provide somewhat privacy protection since the pedestrian section contains few pixels with little information [15]. However, for relatively low-resolution visible images, the pedestrian may be re-identified through deep convolutional neural networks [16]. Subsequently, thermal-infrared imagery was used as a privacy-preserving method since it is difficult to interpret thermal imagery and further identify the object of pedestrian [17]. Nevertheless, pedestrian detection over thermal images collected at daytime is not sufficient. It is essential to offer robust pedestrian detection regardless of time or weather conditions based on both thermal and visible spectrum imagery [18].

In this paper, we propose a Privacy-Preserving Pedestrian Detection scheme (PPPD) based on the vector homeomorphic encryption (VHE) [19]. All the steps of pedestrian detection (e.g., HOG feature extraction, SVM model training, etc.) can be performed over the encrypted pedestrian images. According to the homeomorphism of the ciphertext, the SVM trainer is constructed from the encrypted kernel matrix and the secure HOG extractor. Although all the images are encrypted for privacy preservation, the accuracy of pedestrian detection will not be degraded, as the homomorphic encryption does not alter the extraction and training results. Specifically, our contributions are twofold.

– PPPD is proposed by utilizing VHE for accurate image-based pedestrian detection with privacy preservation. The secure HOG extractor is constructed based on the HOG algorithm over the encrypted pedestrian images, and an encrypted kernel function is used to generate the secure SVM trainer from the secure HOG extractor. Since all the operations are performed over ciphertexts, PPPD is able to protect the raw images and achieve the security of extracted HOG features. In addition, extensive experiments are conducted to demonstrate that PPPD achieves high accuracy of pedestrian detection on multiple pedestrian datasets with lower computation and communication overhead compared with the existing schemes.
– The encrypted inner product operations are designed to allow various kernel functions, such as linear, polynomial and Gaussian kernels, to be calculated in a privacy-preserving fashion. Besides, multiple private linear-transforming matrices are constructed to support arbitrary permutations (e.g., shift and rotation) for the encrypted vectors. These design techniques can be of independent interest to be applied to many other privacy-preserving image classification algorithms.

The remainder of this paper is organized as follows. The related works are given in Section II. We present the system models and design goals in Section III, and review the preliminaries in Section IV. Then, PPPD is proposed in Section V, followed by the security analysis and the performance evaluation in Section IV and Section VII, respectively. Finally, we draw the conclusion in Section VIII.

II. Related Works

Many privacy-preserving and efficient image classification technologies (e.g., pedestrian detection) in the outsourced scenario have been proposed in the literature. Firstly, image owners can disturb data to protect privacy, e.g., anonymity [20] or differential privacy [21], [22]. Specifically, for deep-learning based pedestrian detection, pedestrian images can be blurred or masked to protect privacy [23], [24]. Nonetheless, disturbing may lose valuable information and thus reduce classification accuracy in pedestrian detection [25]. Moreover, it cannot truly ensure privacy due to re-identification attacks [26]. Subsequently, secure multi-party computation (SMC) [27], performing the classification tasks jointly by multiple participants in a secure mode, can also guarantee the privacy of pedestrian images. However, SMC requires numerous interactions between participants [28], which bring heavy communication overhead, thereby not assuring performance in pedestrian detection. Nowadays, deep learning is the most prevalent image classification method. Unfortunately, researches show that depth models are vulnerable to adversarial sample attacks [29]. Such attacks have appeared in image classification [30], which incur serious safety hazards in autonomous driving. On the other side, to prevent privacy leakage, it is widely used to encrypt data before uploading them to the cloud [31]–[34]. In particular, homomorphic encryption provides a possible solution to perform computation over encrypted images for secure image classification. Thus, based on homomorphic encryption, many privacy-preserving image classification algorithms [35]–[37] have been designed

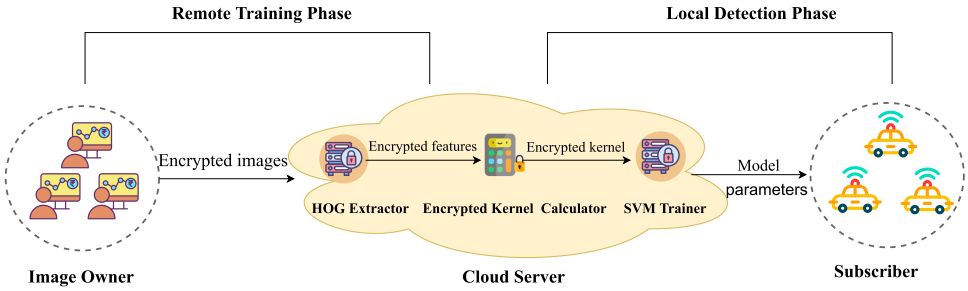


Fig. 2. System model.

for outsourced images processing. However, common homomorphic encryption schemes (e.g., Paillier homomorphic encryption [38]) are inefficiency as images are processed in a pixel-by-pixel way. As a result, an efficient homomorphic encryption scheme supporting pedestrian detection operations over encrypted pedestrian images is urgently needed.

Wang *et al.* [36] gave two privacy-preserving and outsourced HOG algorithms under the single-server and two-server models, respectively. In specific, based on the somewhat homomorphic encryption (SHE) [39], they redesigned each step of the original HOG [8], to transform it into a homomorphic encryption-friendly form. Hence, their algorithms are practically-secure and achieve the approximate performance with the original HOG. Besides, Yang *et al.* [35] and Jiang *et al.* [37] also proposed privacy-preserving image feature extraction schemes to protect the privacy of extracted features. However, subsequent SVM training is still executed in the plaintext domain. In order to protect the image in SVM training, Gonzalez *et al.* [40] presented a privacy-friendly and outsourced SVM training algorithm. Concretely, by leveraging the BCP cryptosystem by Bresson, Catalano and Pointcheval [41], they designed a privacy-preserving primal estimated sub-gradient solver, in which only one comparison is required for each iteration. Therefore, their algorithm can provide good convergence and accuracy. Unfortunately, no privacy protection mechanism is designed to preserve images in HOG feature extraction. To protect the operations of both HOG extraction and SVM training, a trivial solution is to combine the above schemes. Nonetheless, the simple combination would suffer from severe efficiency issues. In this case, the cloud has to return the encrypted HOG features to the image owner for SHE decryption, and the image owner re-encrypts the features using the BCP cryptosystem and uploads ciphertexts to the cloud for SVM training. This, obviously, leads to considerable computation and communication costs, thereby not ensuring performance in pedestrian detection.

As a variant of homomorphic encryption, VHE is capable of encrypting all the elements of a vector in a batching fashion [19]. By leveraging batch encryption, VHE can be used to encrypt the pedestrian images efficiently while supporting secure linear transformations. Hence, by designing each step of HOG to a VHE-friendly form, the privacy protection of both raw images and extracted HOG features can be achieved. Moreover, according to the homomorphism of VHE, the HOG and SVM can be performed in the ciphertext domain. Therefore, the operations from HOG extraction to SVM training can be securely and efficiently performed.

III. System Models and Design Goals

In this section, we give the system model and identify the design goals.

A. System Model

As illustrated in Fig. 2, the system model consists of the following three entities.

- *Image Owner:* The image owner has devices, such as cameras or surveillance systems, to collect a large number of pedestrian images $\boldsymbol{D}$, and intends to provide a high-quality pedestrian detection service. Unfortunately, the owner does not have enough computational resources for image processing. Therefore, the owner seeks help by outsourcing the encrypted pedestrian images $\boldsymbol{D}_e$ and the computational tasks to the cloud.
- *Cloud Server:* The cloud server maintains the pedestrian images and performs the tasks of pedestrian detection. Specifically, the cloud server firstly executes the HOG extraction algorithm on the encrypted pedestrian images $\boldsymbol{D}_e$. Then, leveraging the encrypted extracted features, the cloud server calculates the encrypted kernel matrix, which is used to train the SVM classifier. Finally, the cloud publishes the trained model to the service subscribers. Note that the cloud server also pushes the frequently updated model parameters for the service subscribers to fit different driving environments.
- *Service Subscriber:* Service subscribers pull the trained model from the cloud via secure channels (e.g., SSL). They exploit the model to implement online pedestrian detection locally. Particularly, they also download updated parameters to ensure high flexibility in different environments.

Our system model consists of two phases, remote training and local detection. In remote training, the cloud server extracts the HOG features, generates the kernel matrix, and trains the SVM model based on the maintained pedestrian images. In local detection, the service subscribers use the trained model to accomplish quick pedestrian predictions locally.

In the system model, due to the risks of data leakage in the cloud, the image contents should be protected. Also, the potential attackers may reconstruct the images based on the

TABLE I
NOTATIONS

Notation	Meaning
λ	Security parameter
l	Binary representation parameter
w	Scaling parameter
$\boldsymbol{e}$	Error vector with $\|\boldsymbol{e}\| < w/2$
$\|\boldsymbol{a}\|$	Maximum element in $\boldsymbol{a}$
$\lceil \boldsymbol{a} \rfloor$	Round each element of $\boldsymbol{a}$ to the nearest integer
$\boldsymbol{G}$, $\boldsymbol{M}$	Linear-transforming and key-switching matrices
$\boldsymbol{D}$, $\boldsymbol{D}_e$	Plaintext and ciphertext of pedestrian dataset
$\boldsymbol{K}$, $\boldsymbol{K}_e$	Plaintext and ciphertext of kernel matrix
$\boldsymbol{I}^i(x,y)$, $\boldsymbol{I}_e^i(x,y)$	Plaintext and ciphertext of i-th row of image $\boldsymbol{I}$
$Diff(\alpha)$,$Diff_e(\alpha)$	Plaintext and ciphertext of gradient difference value along α orientation
$Or(\alpha)$, $Or_e(\alpha)$	Plaintext and ciphertext of gradient accumulation along α orientation
$Cell$, $Cell_e$	Plaintext and ciphertext of cell descriptor
$Block$, $Block_e$	Plaintext and ciphertext of block descriptor
HOG, HOG_e	Plaintext and ciphertext of HOG descriptor

extracted image features [42]; thus the privacy of features should be guaranteed. In addition, we assume the security channels have been built between the cloud server and subscribers for the model parameter publication.

B. Design Goals

To build practical pedestrian detection, the following goals should be achieved for our PPPD.

- *Security*. The confidentiality of image contents and extracted feature descriptors should be achieved. Each step of pedestrian detection should be securely performed, i.e., no data leakage will occur in pedestrian detection.
- *Accuracy*. The accuracy of pedestrian detection should be guaranteed over different sources of images. Further, the detection accuracy over encrypted pedestrian images will not be dramatically decreased compared with that over the images directly.
- *Efficiency*. The detection should be efficiently performed on the pedestrian images, even the number of images is large.

IV. PRELIMINARIES

In this section, we review VHE [19], HOG [8] and SVM [43]. The notations are listed in Table I. Here, the lowercase bold and capital bold characters represent vectors and matrices, respectively, e.g., $\boldsymbol{a}$ is a column vector and $\boldsymbol{A}$ is a matrix.

A. Vector Homomorphic Encryption

VHE is composed of the following probabilistic-polynomial-time algorithms [19].

Key Generation (KG): Given a security parameter λ, this algorithm randomly chooses $l, m, n, p, q, w \in \mathbb{Z}$, a noise distribution χ on $\mathbb{Z}_q$, in which $m < n$, $q \gg p$, $w(p-1) < q$, and $l = \lceil \log_2 (q-1) \rfloor$. Then, it generates a matrix $\boldsymbol{S} = [\boldsymbol{I}, \boldsymbol{T}]$, where $\boldsymbol{I}$ is an identity matrix of $m \times m$ and $\boldsymbol{T}$ is a random matrix of $m \times (n-m)$. Finally, the algorithm keeps the key $\boldsymbol{S}$ secretly and publishes the pubic parameter $Param = (l, m, n, p, q, w, \chi)$.

Encryption (Enc): Given a plaintext vector $\boldsymbol{x} \in \mathbb{Z}_p^m$ and the secret key $\boldsymbol{S} \in \mathbb{Z}^{m \times n}$, this algorithm generates a corresponding ciphertext vector $\boldsymbol{c} \in \mathbb{Z}_q^n$ satisfying the following equation

$$\boldsymbol{S}\boldsymbol{c} = w\boldsymbol{x} + \boldsymbol{e}, \tag{1}$$

where $\boldsymbol{e}$ is a small error vector.

Decryption (Dec): Given a ciphertext vector $\boldsymbol{c} \in \mathbb{Z}_q^n$ and the secret key $\boldsymbol{S} \in \mathbb{Z}^{m \times n}$, this algorithm recovers a corresponding plaintext vector $\boldsymbol{x} \in \mathbb{Z}_p^m$ satisfying the following equation

$$\boldsymbol{x} = \lceil \frac{\boldsymbol{S}\boldsymbol{c}}{w} \rfloor_q. \tag{2}$$

Key Switching (KS): By using a key-switching matrix $\boldsymbol{M}$, this algorithm converts an old key/ciphertext pair $(\boldsymbol{S}_{old}, \boldsymbol{c}_{old})$ to a new pair $(\boldsymbol{S}_{new}, \boldsymbol{c}_{new})$ with the same plaintext $\boldsymbol{x}$. We have $\boldsymbol{c}_{new} = \boldsymbol{M}\boldsymbol{c}_{old}$ and $\boldsymbol{S}_{new}\boldsymbol{c}_{new} = \boldsymbol{S}_{old}\boldsymbol{c}_{old} = w\boldsymbol{x} + \boldsymbol{e}$. $VHE.KS$ can be expressed as

$$(\boldsymbol{M}, \boldsymbol{S}_{new}) \leftarrow VHE.KS(\boldsymbol{S}_{old}). \tag{3}$$

Addition (Add): For plaintexts $\boldsymbol{x_1}, \boldsymbol{x_2}$ and corresponding ciphertexts $\boldsymbol{c_1}, \boldsymbol{c_2}$ that are encrypted with the same secret key $\boldsymbol{S}$, we have

$$\boldsymbol{S}(\boldsymbol{c_1} + \boldsymbol{c_2}) = w(\boldsymbol{x_1} + \boldsymbol{x_2}) + (\boldsymbol{e_1} + \boldsymbol{e_2}). \tag{4}$$

Linear Transformation (LT): Given a plaintext $\boldsymbol{x}$ and its corresponding ciphertext $\boldsymbol{c}$ with the secret key $\boldsymbol{S}$, the linear transformation $\boldsymbol{G}\boldsymbol{x}$ can be implemented as

$$(\boldsymbol{G}\boldsymbol{S})\boldsymbol{c} = w(\boldsymbol{G}\boldsymbol{x}) + \boldsymbol{G}\boldsymbol{e}, \tag{5}$$

where $\boldsymbol{G}$ is a linear-transforming matrix. Thus, $\boldsymbol{c}$ can be treated as the encryption of $\boldsymbol{G}\boldsymbol{x}$ with the secret key $\boldsymbol{G}\boldsymbol{S}$. Note that the subsequent HOG algorithm will perform multiple different linear-transforming matrices $\boldsymbol{G}_i$. To continue to do addition operations, e.g., $\sum_i \boldsymbol{G}_i\boldsymbol{x}_i$, these secret keys $\boldsymbol{G}_i\boldsymbol{S}$ are switched to a common secret key, e.g., the original $\boldsymbol{S}$, by using the corresponding key-switching matrix $\boldsymbol{M}_i$.

Inner Product (IP): Suppose $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are encrypted into $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ with the secret keys $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$, respectively. We have $\boldsymbol{S}_i\boldsymbol{c}_i = w\boldsymbol{x}_i + \boldsymbol{e}_i$, $i = 1, 2$. Further, given a weight matrix $\boldsymbol{H}$, the weighted inner product $\boldsymbol{x}_1^T\boldsymbol{H}\boldsymbol{x}_2$ in the encrypted domain can be calculated as

$$vec(\boldsymbol{S}_1^T\boldsymbol{H}\boldsymbol{S}_2)^T - \left\lfloor \frac{vec(\boldsymbol{c}_1\boldsymbol{c}_2^T)}{w} \right\rfloor = w(\boldsymbol{x}_1^T\boldsymbol{H}\boldsymbol{x}_2) + \boldsymbol{e}, \tag{6}$$

where $vec(\cdot)$ is a vectorization function. Note that, $\boldsymbol{H}$ is an identity matrix in PPPD, and thus the inner product can be represented as $VHE.IP(\boldsymbol{c}_1, \boldsymbol{c}_2)$.

B. Histogram of Oriented Gradients

The original HOG was firstly proposed by Dalal and Triggs [8]. The HOG feature descriptor can maintain invariance to both geometric and optical deformation of images, and thus it is suitable for pedestrian detection. Assume that an image $\boldsymbol{I}$ is represented as a matrix $\boldsymbol{I}(x, y)$ with the size of $n \times m$. The extraction of HOG is described in the following steps.

Image Preprocessing: For the HOG descriptor in common, the first step is the gamma and color normalizations. We skip this step due to the limited performance gain.

Gradient Calculation: The 1-D centered, point discrete derivative mask $[-1, 0, 1]$ in both horizontal and vertical directions is executed for gradient calculation. Specifically, the horizontal and vertical gradients can be calculated as $g_x = \boldsymbol{I}(x+1, y) - \boldsymbol{I}(x-1, y)$ and $g_y = \boldsymbol{I}(x, y+1) - \boldsymbol{I}(x, y-1)$, respectively. Also, the gradient magnitude and direction can be calculated as $g = \sqrt{g_x^2 + g_y^2}$ and $\theta = arctan\frac{g_y}{g_x}$, respectively.

Cell Histogram Building: This image is divided into many small connected regions called cells, and a cell histogram is built for each cell based on the values calculated in the previous step of gradient calculation. Concretely, each pixel within the cell contributes a weighted vote for an orientation-based histogram bin, and the votes are added up into orientation bins. Note that the unsigned orientation bins are evenly spaced over $[0^{\circ}, 180^{\circ}]$. Especially, a bin is selected based on the direction, and the vote is selected based on the magnitude.

Block Normalization: To further weaken the interference of illumination and shadowing, the gradients need to be locally normalized, which requires grouping the cells together into larger, spatially connected blocks. The widely used method of normalization is to exploring $L_2 - norm$, which is defined as $\|\boldsymbol{v}\|_2$. Specifically, suppose that $\boldsymbol{v}$ is the unnormalized block descriptor vector, then the normalization procedure of each block descriptor vector with $L_2 - norm$ is represented as $\frac{\boldsymbol{v}}{\sqrt{\|\boldsymbol{v}\|_2^2 + \epsilon}}$, where ϵ is a small constant.

HOG Descriptor Generation: By concatenating the normalized block descriptors from left to right and from top to bottom for the full image, the final HOG descriptor is generated. Note that these blocks typically overlap, meaning that each cell contributes more than once to the final descriptor.

The HOG descriptor used for pedestrian detection is usually calculated on a 128×64 patch of an image. An image patch with other sizes should be resized to 128×64. We set that a cell contains 8×8 pixels and a block contains 2×2 cells, which is a typical setting for HOG.

C. Support Vector Machine

We consider the two-class SVM problem, in which the separation hyperplane can be solved by correctly separating the training data set with the largest geometric interval. Given a training data set with m sample points $\boldsymbol{A} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$, where $\boldsymbol{x}_i \in \mathbb{R}^n$ is the i-th feature vector, and $y_i \in \{+1, -1\}$ is the class label corresponding to $\boldsymbol{x}_i$ for $i = 1, 2, \ldots, m$. If the training data set is linearly separable, this SVM problem can be transformed into a convex quadratic programming problem as

$$\begin{aligned} \min_{\boldsymbol{w}, b, \xi} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{m}\xi_i \\ \text{s.t.} \quad & y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1 - \xi_i,\ \xi_i \geq 0,\ i = 1, \ldots, m, \end{aligned}$$

where $\boldsymbol{w}$ is the hyperplane normal vector, b is the offset constant, and C is the penalty coefficient.

Lagrangian Dual: The Lagrangian method is used to transform the original SVM problem into a dual problem and find the optimal solution. The objective function can be transformed into:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j(\boldsymbol{x}_i \cdot \boldsymbol{x}_j) - \sum_{i=1}^{m}\alpha_i \\ \text{s.t.} \quad & \sum_{i=1}^{m}\alpha_i y_i = 0,\ 0 \leq \alpha_i \leq C,\ i = 1, \ldots, m, \end{aligned}$$

where α_i is the Lagrangian operator corresponding to the data sample point $\{\boldsymbol{x}_i, y_i\}$, for $i = 1, 2, \ldots, m$.

Kernel Technique: As the training data set is not always linearly separable, we convert a non-linear classification problem to a linear problem by using kernel techniques. Specifically, the objective function can be further transformed as

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{i=1}^{m}\alpha_i \\ \text{s.t.} \quad & \sum_{i=1}^{m}\alpha_i y_i = 0,\ 0 \leq \alpha_i \leq C, i = 1, \ldots, m, \end{aligned}$$

where $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is a kernel function, mainly including linear, polynomial, and Gaussian kernels.

Sequential minimal optimization (SMO): SMO is a typical SVM training algorithm [43]. In SMO, all variables satisfy the KKT condition of this optimization problem. SMO consists of two parts, an analytical method that solves the quadratic programming of two variables, and a heuristic method that is used to select variables.

V. Proposed PPPD

In this section, we propose the PPPD scheme, which involves three modules.

A. Framework of PPPD

As illustrated in Fig. 3, the image owner first encrypts the pedestrian images before uploading them to the cloud server. Then, the cloud server builds the training model through three modules. Finally, the service subscribers, e.g., the self-driving vehicles, obtain the model parameters via secure channels and perform pedestrian detection. This step executed locally on the service subscribers, is the same as that in the traditional pedestrian detection over plaintexts; thus, we omit the description of this step.

- *Secure HOG Extractor:* Upon receiving the encrypted pedestrian images $\boldsymbol{D}_e$ and the constructed key-switching matrices $\boldsymbol{M}_i$ corresponding to the linear-transforming matrices $\boldsymbol{G}_i$, the cloud server calculates the gradient values and creates the encrypted HOG descriptors HOG_e. Note that the meanings of $\boldsymbol{G}_i$ and $\boldsymbol{M}_i$ are given in Table II and the constructions of these matrices will be involved later.

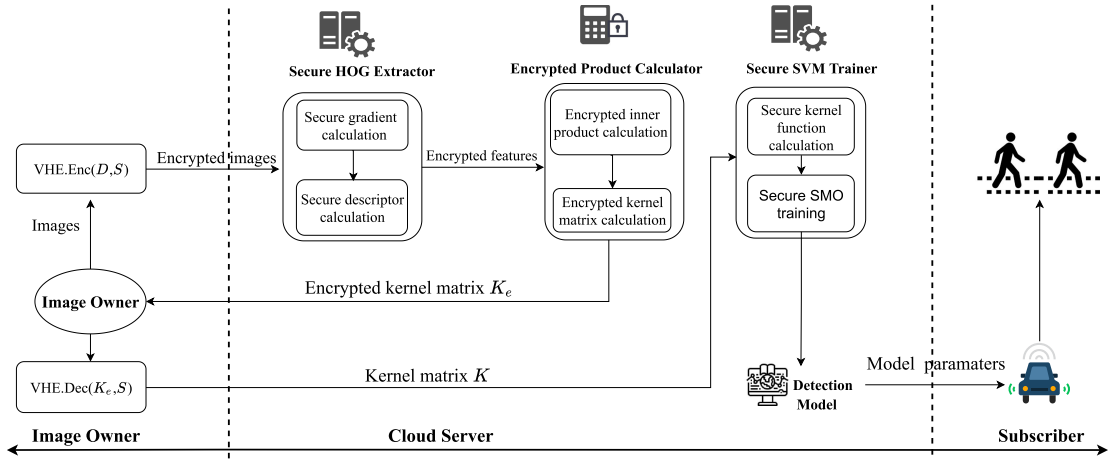


Fig. 3. Framework of PPPD.

TABLE II
MEANINGS OF $\boldsymbol{G}_i$ AND $\boldsymbol{M}_i$

$\boldsymbol{G}_i$	$\boldsymbol{M}_i$	Meaning
$\boldsymbol{G}_L, \boldsymbol{G}_R$	$\boldsymbol{M}_L, \boldsymbol{M}_R$	For vector left (right)
$\boldsymbol{G}_\alpha$	$\boldsymbol{M}_\alpha$	For gradient calculation along $\alpha(= 0°, 45°, 90°, 135°)$
$\boldsymbol{G}_H$	$\boldsymbol{M}_H$	For cell descriptor building
$\boldsymbol{G}_{up}, \boldsymbol{G}_{down}$	$\boldsymbol{M}_{up}, \boldsymbol{M}_{down}$	For Block descriptor building

– *Encrypted Product Calculator:* The encrypted kernel matrix $\boldsymbol{K}_e$ is first calculated upon HOG_e by the cloud server through the inner product supported by VHE. Then, $\boldsymbol{K}_e$ is returned to the image owner for decryption to acquire the kernel matrix $\boldsymbol{K}$. Finally, $\boldsymbol{K}$ is uploaded to the cloud for the SMO training.

– *Secure SVM Trainer:* Upon receiving $\boldsymbol{K}$, the SMO algorithm is executed by the cloud server to train the pedestrian detection model. Eventually, the trained model is pushed to subscribers for local detection. Besides, the cloud server publishes updated parameters to the service subscribers for pedestrian detection in different environments.

B. Secure HOG Extractor

As VHE does not support all operations of the original plaintext HOG algorithm, every step of HOG is required to be carefully re-designed to adapt it into a VHE-friendly form. The building of the secure HOG extractor contains two stages: 1) Re-designing the HOG algorithm in the plaintext domain, and 2) Converting the re-designed algorithm into the ciphertext form.

1) Re-Designed HOG Over Plaintexts: VHE processes vectors supporting the homomorphic operations of linear transformation and inner product. The re-designing HOG should be compatible with the merits of VHE. The HOG algorithm over plaintexts is re-designed following the steps below.

Step 1. Image Vectorization: An image $\boldsymbol{I}_{n\times m}$ can be represented as an integer matrix $\boldsymbol{I}(x, y)$ with the size of $n \times m$, and each row of $\boldsymbol{I}$ can be treated as an m-dimensional integer vector. Thus, $\boldsymbol{I}$ is divided row-by-row into a group of vectors as $\{\boldsymbol{I}^1(x, y), \ldots, \boldsymbol{I}^n(x, y)\}$.

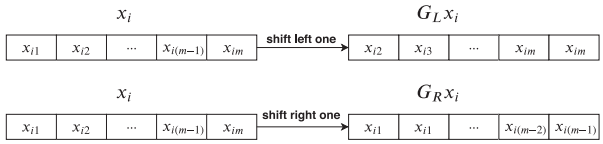


Fig. 4. Vector horizontal shift.

Step 2. Shift Operations design: To manipulate effectively rows of $\boldsymbol{I}$, gradient calculations are all carried out over *vector horizontal shift* and *vector vertical shift* in the following way.

– *Vector horizontal shift.* The pixel-wise vector shift operation is the right (left) shift, which moves the pixels of a vector to the right (left) by one pixel, as shown in Fig. 4. The pixel position vacated by shift is proximity-filled, and the pixel shifted off the end is discarded. Then, to adopt the vector shift operations, two transforming matrices $\boldsymbol{G}_L, \boldsymbol{G}_R \in \mathbb{Z}_2^{m\times m}$ are constructed.

$$\boldsymbol{G}_L = \begin{bmatrix} 0 & 1 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 \\ 0 & 0 & \cdots & 1 \end{bmatrix}, \boldsymbol{G}_R = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}.$$

Thus, we have the right shift as

$$\boldsymbol{I}^i(x-1, y) = \boldsymbol{G}_R \boldsymbol{I}^i(x, y), \tag{7}$$

and the left shift as

$$\boldsymbol{I}^i(x+1, y) = \boldsymbol{G}_L \boldsymbol{I}^i(x, y). \tag{8}$$

– *Vector vertical shift:* The vertical shift is a simple transition between two adjacent rows. We have the up shift as

$$\boldsymbol{I}^i(x, y-1) = \boldsymbol{I}^{i-1}(x, y), \tag{9}$$

and the down shift as

$$\boldsymbol{I}^i(x, y+1) = \boldsymbol{I}^{i+1}(x, y). \tag{10}$$

Step 3. Gradient Calculation: In gradient calculation, the technique in [36] is adopted to improve the efficiency of the original HOG algorithm. To be specific, the number of the original orientations is reduced to be four (i.e., $0^{\circ}, 45^{\circ}, 90^{\circ}, 135^{\circ}$). Then, the oriented gradients are calculated in the operating unit of pixels as

$$\begin{cases} Diff(0^{\circ}) = \boldsymbol{I}(x+1,y) - \boldsymbol{I}(x-1,y) \\ Diff(45^{\circ}) = \boldsymbol{I}(x-1,y-1) - \boldsymbol{I}(x+1,y+1) \\ Diff(90^{\circ}) = \boldsymbol{I}(x,y+1) - \boldsymbol{I}(x,y-1) \\ Diff(135^{\circ}) = \boldsymbol{I}(x+1,y-1) - \boldsymbol{I}(x-1,y+1) \end{cases} \tag{11}$$

Further, by using the designed shift operations, these gradients can be calculated in the operating unit of rows as

$$\begin{cases} Diff^{i}(0^{\circ}) = \boldsymbol{G}_L \boldsymbol{I}^{i}(x,y) - \boldsymbol{G}_R \boldsymbol{I}^{i}(x,y) \\ Diff^{i}(45^{\circ}) = \boldsymbol{G}_R \boldsymbol{I}^{i-1}(x,y) - \boldsymbol{G}_L \boldsymbol{I}^{i+1}(x,y) \\ Diff^{i}(90^{\circ}) = \boldsymbol{I}^{i+1}(x,y) - \boldsymbol{I}^{i-1}(x,y) \\ Diff^{i}(135^{\circ}) = \boldsymbol{G}_L \boldsymbol{I}^{i-1}(x,y) - \boldsymbol{G}_R \boldsymbol{I}^{i+1}(x,y) \end{cases} \tag{12}$$

Step 4. Cell Histogram Building: According to the typical setting of HOG, a cell contains 8×8 pixels. In a cell, four orientations ($0^{\circ}, 45^{\circ}, 90^{\circ}, 135^{\circ}$) of the 8×8 pixels are further accumulated to obtain the cell histogram descriptor. Nevertheless, the image $\boldsymbol{I}_{n\times m}$ is now denoted as n vectors of m dimensions. Therefore, the feature in one cell cannot be separately calculated in the operating unit of pixels, but in the operating unit of rows. Specifically, to obtain the cell descriptor with 8×8 pixels, two cumulations of gradients for the horizontal 8 pixels and the vertical 8 pixels are performed, respectively.

– *Horizontal cumulation of gradients:* To perform such operation in the unit of rows, a linear-transforming matrix $\boldsymbol{G}_H \in \mathbb{Z}_2^{8\times m}$ is constructed for the cumulative sum of gradients of adjacent horizontal 8 pixels as

$$\boldsymbol{G}_H = \begin{bmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_0 & \cdots & \boldsymbol{x}_0 \\ \boldsymbol{x}_0 & \boldsymbol{x}_1 & \cdots & \boldsymbol{x}_0 \\ \cdots & \cdots & \cdots & \cdots \\ \boldsymbol{x}_0 & \boldsymbol{x}_0 & \cdots & \boldsymbol{x}_1 \end{bmatrix},$$

where $\boldsymbol{x}_0 = [00\,000\,000]$ and $\boldsymbol{x}_1 = [11\,111\,111]$.

– *Vertical cumulation of gradients:* To take the cumulation along $\alpha(= 0^{\circ}, 45^{\circ}, 90^{\circ}, 135^{\circ})$ in a cell, by adding 8 consecutive rows, such cumulation can be achieved in the j-cell (its first row is the i-row of $\boldsymbol{I}$) as

$$Or^{j}(\alpha) = \boldsymbol{G}_H \sum_{k=0}^{7} Diff^{i+k}(\alpha). \tag{13}$$

Note that the four direction gradients need to be joined in a cell. That is, the row vectors in four directions are shifted by linear transformation, and then added in series to obtain the cell descriptor. To this end, the following four linear-transforming matrices, $\boldsymbol{G}_0, \boldsymbol{G}_{45}, \boldsymbol{G}_{90}, \boldsymbol{G}_{135} \in \mathbb{Z}_2^{32\times 8}$, are constructed as

$$\boldsymbol{G}_0 = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \end{bmatrix}, \boldsymbol{G}_{45} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 1 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 \end{bmatrix},$$

$$\boldsymbol{G}_{90} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ 1 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 \\ 0 & 0 & \cdots & 0 \end{bmatrix}, \boldsymbol{G}_{135} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ 1 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}.$$

Finally, by using the four linear-transforming matrices, the j-th cell descriptor can be derived as

$$\begin{aligned} Cell^{j} = {} & \boldsymbol{G}_0 Or^{j}(0^{\circ}) + \boldsymbol{G}_{45} Or^{j}(45^{\circ}) \\ & + \boldsymbol{G}_{90} Or^{j}(90^{\circ}) + \boldsymbol{G}_{135} Or^{j}(135^{\circ}). \end{aligned} \tag{14}$$

Step 5. Block and HOG Descriptors Generation: In a typical setting, a block contains 2×2 cells. Thus, a block descriptor can be generated by the linear transformation on two rows of cell descriptors. Two linear-transforming matrices, $\boldsymbol{G}_{up}, \boldsymbol{G}_{down} \in \mathbb{Z}_2^{16\times[\frac{n}{16}-1]\times 32}$, are constructed to set relative positions of such 2×2 cell descriptors in the block descriptor.

$$\boldsymbol{G}_{up} = \begin{bmatrix} \boldsymbol{E}_4 & \boldsymbol{E}_0 & \boldsymbol{E}_0 & \boldsymbol{E}_0 & \cdots & \boldsymbol{E}_0 & \boldsymbol{E}_0 \\ \boldsymbol{E}_0 & \boldsymbol{E}_4 & \boldsymbol{E}_0 & \boldsymbol{E}_0 & \cdots & \boldsymbol{E}_0 & \boldsymbol{E}_0 \\ \boldsymbol{E}_0 & \boldsymbol{E}_0 & \boldsymbol{E}_0 & \boldsymbol{E}_0 & \cdots & \boldsymbol{E}_0 & \boldsymbol{E}_0 \\ \boldsymbol{E}_0 & \boldsymbol{E}_0 & \boldsymbol{E}_0 & \boldsymbol{E}_0 & \cdots & \boldsymbol{E}_0 & \boldsymbol{E}_0 \\ \boldsymbol{E}_0 & \boldsymbol{E}_4 & \boldsymbol{E}_0 & \boldsymbol{E}_0 & \cdots & \boldsymbol{E}_0 & \boldsymbol{E}_0 \\ \boldsymbol{E}_0 & \boldsymbol{E}_0 & \boldsymbol{E}_4 & \boldsymbol{E}_0 & \cdots & \boldsymbol{E}_0 & \boldsymbol{E}_0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \boldsymbol{E}_0 & \boldsymbol{E}_0 & \boldsymbol{E}_0 & \boldsymbol{E}_0 & \cdots & \boldsymbol{E}_0 & \boldsymbol{E}_0 \end{bmatrix},$$

$$\boldsymbol{G}_{down} = \begin{bmatrix} \boldsymbol{E}_0 & \boldsymbol{E}_0 & \boldsymbol{E}_0 & \boldsymbol{E}_0 & \cdots & \boldsymbol{E}_0 & \boldsymbol{E}_0 \\ \boldsymbol{E}_0 & \boldsymbol{E}_0 & \boldsymbol{E}_0 & \boldsymbol{E}_0 & \cdots & \boldsymbol{E}_0 & \boldsymbol{E}_0 \\ \boldsymbol{E}_4 & \boldsymbol{E}_0 & \boldsymbol{E}_0 & \boldsymbol{E}_0 & \cdots & \boldsymbol{E}_0 & \boldsymbol{E}_0 \\ \boldsymbol{E}_0 & \boldsymbol{E}_4 & \boldsymbol{E}_0 & \boldsymbol{E}_0 & \cdots & \boldsymbol{E}_0 & \boldsymbol{E}_0 \\ \boldsymbol{E}_0 & \boldsymbol{E}_0 & \boldsymbol{E}_0 & \boldsymbol{E}_0 & \cdots & \boldsymbol{E}_0 & \boldsymbol{E}_0 \\ \boldsymbol{E}_0 & \boldsymbol{E}_0 & \boldsymbol{E}_0 & \boldsymbol{E}_0 & \cdots & \boldsymbol{E}_0 & \boldsymbol{E}_0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \boldsymbol{E}_0 & \boldsymbol{E}_0 & \boldsymbol{E}_0 & \boldsymbol{E}_0 & \cdots & \boldsymbol{E}_0 & \boldsymbol{E}_4 \end{bmatrix},$$

where $\boldsymbol{E}_4$ and $\boldsymbol{E}_0$ represent 4×4 identity and all-zero matrices, respectively. As a result, the k-th block descriptor is built as

$$Block^{k} = \boldsymbol{G}_{up} Cell^{j} + \boldsymbol{G}_{down} Cell^{j+1}. \tag{15}$$

To create the final HOG descriptor of an image $\boldsymbol{I}$, a sliding window with 32×32 pixels (i.e., the block size) is used to scan $\boldsymbol{I}$

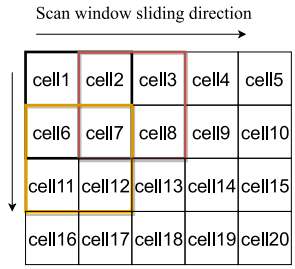


Fig. 5. Sliding window scanning.

from top to bottom and from left to right with the step of 8 pixels, as illustrated in Fig. 5. As the scanned blocks overlap, there are $(\frac{m}{8}-1)$ and $(\frac{n}{8}-1)$ blocks in the horizontal and vertical directions, respectively. Therefore, for $\boldsymbol{I}_{n\times m}$, there are $(\frac{m}{8}-1)\times(\frac{n}{8}-1)$ blocks. Further, since the calculations are executed in units of row, there are totally $k=\frac{n}{8}-1$ rows, and each row contains $\frac{m}{8}-1$ blocks. Consequently, the final HOG descriptor is generated as

$$HOG=\{Block^1, Block^2, \ldots, Block^k\}. \tag{16}$$

2) Re-Designed HOG Over Ciphertexts: According to the designed HOG over plaintexts, the corresponding HOG over ciphertexts is generated by firstly encrypting each row of an image, and then switching linear-transforming matrices $\boldsymbol{G}_i$ (e.g., $\boldsymbol{G}_L$) in the plaintext domain into key-switching matrices $\boldsymbol{M}_i$ (e.g., $\boldsymbol{M}_L$) in the ciphertext domain. In specific, the re-designed HOG over ciphertexts is obtained in the following steps.

Step 1. System initialization: The system initialization is carried out by the image owner, firstly by invoking $Param \leftarrow VHE.KG(\lambda)$ and $(\boldsymbol{M}_i, \boldsymbol{S}) \leftarrow VHE.KS(\boldsymbol{G}_i\boldsymbol{S})$, then uploading $Param$ and $\boldsymbol{M}_i$ to the cloud.

Taking the linear transformation operation $\boldsymbol{Gx}$ as an example, we show how to design the corresponding key switching matrix $\boldsymbol{M}$ to satisfy the condition that $\boldsymbol{c}_{new}=\boldsymbol{Mc}^*$ is the encryption of $\boldsymbol{Gx}$ under the new secret key $\boldsymbol{S}_{new}$, where $\boldsymbol{G}$ is the linear transformation matrix, $\boldsymbol{x}$ is the plaintext vector, and $\boldsymbol{c}$ is the ciphertext of $\boldsymbol{x}$ under the secret key $\boldsymbol{S}$. Note that $\boldsymbol{c}^*$ and $(\boldsymbol{GS})^*$ are the binary representations of the vector $\boldsymbol{c}$ and the matrix $\boldsymbol{GS}$, respectively. We have $(\boldsymbol{GS})^*\boldsymbol{c}^*=(\boldsymbol{GS})\boldsymbol{c}$. Also, we have $\boldsymbol{Sc}=w\boldsymbol{x}+\boldsymbol{e}$, where w is the scaling parameter, and $\boldsymbol{e}$ is the small noise vector.

We first generate the new secret key as $\boldsymbol{S}_{new}=[\boldsymbol{I}, \boldsymbol{T}']$, where $\boldsymbol{I}$ is an identity matrix, and $\boldsymbol{T}'$ is a random matrix. Then, the corresponding key switching matrix $\boldsymbol{M}$ of $\boldsymbol{Gx}$ can be calculated as

$$\boldsymbol{M}=\begin{bmatrix}(\boldsymbol{GS})^*-\boldsymbol{T}'\boldsymbol{A}+\boldsymbol{E}\\ \boldsymbol{A}\end{bmatrix},$$

where $\boldsymbol{A}$ is a random matrix, and $\boldsymbol{E}$ is a small noise matrix.

In this case, by using $\boldsymbol{S}_{new}$ to decrypt $\boldsymbol{c}_{new}$, we have

$$\begin{aligned}\boldsymbol{S}_{new}\boldsymbol{c}_{new}&=[\boldsymbol{I},\boldsymbol{T}']\begin{bmatrix}(\boldsymbol{GS})^*-\boldsymbol{T}'\boldsymbol{A}+\boldsymbol{E}\\ \boldsymbol{A}\end{bmatrix}\boldsymbol{c}^*\\ &=(\boldsymbol{GS})^*\boldsymbol{c}^*+\boldsymbol{Ec}^*\\ &=\boldsymbol{GSc}+\boldsymbol{Ec}^*\\ &=\boldsymbol{G}(w\boldsymbol{x}+\boldsymbol{e})+\boldsymbol{Ec}^*\\ &=w(\boldsymbol{Gx})+\boldsymbol{Ge}+\boldsymbol{Ec}^*\\ &=w(\boldsymbol{Gx})+\boldsymbol{e}_{new}\end{aligned}$$

where $\boldsymbol{e}_{new}=\boldsymbol{Ge}+\boldsymbol{Ec}^*$ is a new small noise vector.

Therefore, $\boldsymbol{c}_{new}$ is the corresponding ciphertext vector of $\boldsymbol{Gx}$.

Step 2. Image Encryption: The image $\boldsymbol{I}_{n\times m}$ is encrypted with a secret key $\boldsymbol{S}$ row-by-row as

$$\boldsymbol{I}_e^i(x,y)=VHE.Enc(\boldsymbol{I}^i(x,y),\boldsymbol{S}), \tag{17}$$

where $\boldsymbol{I}_e^i(x,y)$ is the encryption of $\boldsymbol{I}^i(x,y)$. Finally, the encrypted image, denoted as $\boldsymbol{I}_e=\{\boldsymbol{I}_e^1(x,y),\ldots,\boldsymbol{I}_e^n(x,y)\}$, is uploaded to the cloud.

Step 3. Secure Gradient Calculation: Utilizing the linear-transforming matrices, the gradients can be calculated in the ciphertext domain by the cloud. Since $\boldsymbol{I}_e^i(x,y)$ is the ciphertext of $\boldsymbol{I}^i(x,y)$ with $\boldsymbol{S}$, we have $\boldsymbol{S}\boldsymbol{I}_e^i(x,y)=w\boldsymbol{I}^i(x,y)+\boldsymbol{e}$, where $\boldsymbol{e}$ is a small noise vector. Then, we obtain

$$\begin{cases}(\boldsymbol{G}_L\boldsymbol{S})\boldsymbol{I}_e^i(x,y)=w(\boldsymbol{G}_L\boldsymbol{I}^i(x,y))+\boldsymbol{G}_L\boldsymbol{e}\\ (\boldsymbol{G}_R\boldsymbol{S})\boldsymbol{I}_e^i(x,y)=w(\boldsymbol{G}_R\boldsymbol{I}^i(x,y))+\boldsymbol{G}_R\boldsymbol{e}\end{cases} \tag{18}$$

The ciphertexts of $\boldsymbol{G}_L\boldsymbol{I}^i(x,y)$ and $\boldsymbol{G}_R\boldsymbol{I}^i(x,y)$, in the same $\boldsymbol{I}_e^i(x,y)$, are generated with the secret keys $\boldsymbol{G}_L\boldsymbol{S}$ and $\boldsymbol{G}_R\boldsymbol{S}$, respectively. To perform homomorphic addition, $\boldsymbol{G}_L\boldsymbol{S}$ and $\boldsymbol{G}_R\boldsymbol{S}$ need to be switched into the same secret key $\boldsymbol{S}$. To be specific, key-switching matrices $\boldsymbol{M}_L$ and $\boldsymbol{M}_R$ are generated to convert $\boldsymbol{G}_L\boldsymbol{S}$ and $\boldsymbol{G}_R\boldsymbol{S}$ to $\boldsymbol{S}$. Then, the encrypted gradients are obtained as

$$\begin{cases}Diff_e^i(0^\circ)=\boldsymbol{M}_L\boldsymbol{I}_e^i(x,y)-\boldsymbol{M}_R\boldsymbol{I}_e^i(x,y)\\ Diff_e^i(45^\circ)=\boldsymbol{M}_R\boldsymbol{I}_e^{i-1}(x,y)-\boldsymbol{M}_L\boldsymbol{I}_e^{i+1}(x,y)\\ Diff_e^i(90^\circ)=\boldsymbol{I}_e^{i+1}(x,y)-\boldsymbol{I}_e^{i-1}(x,y)\\ Diff_e^i(135^\circ)=\boldsymbol{M}_L\boldsymbol{I}_e^{i-1}(x,y)-\boldsymbol{M}_R\boldsymbol{I}_e^{i+1}(x,y)\end{cases} \tag{19}$$

Step 4. Secure HOG Descriptor Generation: To take the accumulation along the direction of $\alpha(=0^\circ, 45^\circ, 90^\circ$ and $135^\circ)$ in a cell over ciphertexts, the cloud firstly calculates the new ciphertext $\boldsymbol{M}_H Diff_e^i(\alpha)$ representing the encryption of the sum of every 8-pixel gradients in the i-th row, where $(\boldsymbol{M}_H, \boldsymbol{S})\leftarrow VHE.KS(\boldsymbol{G}_H\boldsymbol{S})$. Then, by adding 8 consecutive rows of gradient ciphertexts, the cloud obtains the accumulation along α in a cell as

$$Or(\alpha)_e^j=\boldsymbol{M}_H\left(Diff_e^i(\alpha)+\cdots+Diff_e^{i+7}(\alpha)\right). \tag{20}$$

Similarly, the encrypted gradient vectors of four directions in a cell are required to be joined in sequence. $\boldsymbol{G}_0$, $\boldsymbol{G}_{45}$, $\boldsymbol{G}_{90}$ and $\boldsymbol{G}_{135}$ are converted into $\boldsymbol{M}_0$, $\boldsymbol{M}_{45}$, $\boldsymbol{M}_{90}$ and $\boldsymbol{M}_{135}$, respectively. Thus, a row of cell descriptor over ciphertexts is derived as

$$\begin{aligned}Cell_e^j=&\boldsymbol{M}_0Or(0^\circ)_e^j+\boldsymbol{M}_{45}Or(45^\circ)_e^j\\ &+\boldsymbol{M}_{90}Or(90^\circ)_e^j+\boldsymbol{M}_{135}Or(135^\circ)_e^j.\end{aligned} \tag{21}$$

Further, the block descriptor is obtained over ciphertexts as

$$Block_e^k=\boldsymbol{M}_{up}Cell_e^j+\boldsymbol{M}_{down}Cell_e^{j+1}. \tag{22}$$

Algorithm 1: $\boldsymbol{K}_e$ Calculation.

Input: n ciphertext HOG features matrix $\{HOG_e^1, HOG_e^2, \ldots, HOG_e^n\}$
Output: ciphertext kernel matrix $\boldsymbol{K}_e^{n\times n}$

```
set K_e[n, n] empty
for i=1 to n do
  for j=1 to n do
    for q=1 to k do
      calculate inner product: K_e[i, j]+ = VHE.IP(HOG_e^i.Block_e^q, HOG_e^j.Block_e^q)
    end for
  end for
end for
return K_e
```

Algorithm 2: Improved $\boldsymbol{K}_e$ Calculation.

Input: n ciphertext HOG features matrix $\{HOG_e^1, HOG_e^2, \ldots, HOG_e^n\}$
Output: ciphertext kernel matrix $\boldsymbol{K}_e^{n\times n}$

```
set K_e[n, n] and Tmp_e[k] empty
for q=1 to k do
  Tmp_e[q]+ = VHE.Add(HOG^q.Block^1, ..., HOG^q.Block^k)
end for
for i=1 to n do
  for j=1 to n do
    K_e[i, j]+ = VHE.IP(Tmp_e^i, Tmp_e^j)
  end for
end for
return K_e
```

Finally, the final HOG descriptor in the encrypted domain is created as

$$HOG_e = \{Block_e^1, Block_e^2, \ldots, Block_e^k\}. \tag{23}$$

C. Encrypted Product Calculator

After the HOG descriptors are extracted from ciphertext images, the ciphertext kernel matrix $\boldsymbol{K}_e$ is constructed by calculating inner product in the encrypted domain, and decrypted by the image owner to obtain the kernel matrix $\boldsymbol{K}$.

1) Ciphertext Kernel Matrix Construction: Each of n encrypted HOG descriptors $\{HOG_e^1, HOG_e^2, \ldots, HOG_e^n\}$ consists of k encrypted block descriptors. The ciphertext kernel matrix can be calculated via Algorithm 1 and Algorithm 2.

- In Algorithm 1, inner products are calculated and the results are added up, which consume $n^2 \times k$ homomorphic operations.
- In Algorithm 2, k blocks of each HOG descriptor are added up and then subjected to subsequent inner product operations. Only n^2 inner products are calculated.

As a consequence, Algorithm 2 saves an approximately k times of computation and communication consumptions compared with Algorithm 1.

2) Ciphertext Kernel Matrix Decryption: Upon receiving the ciphertext kernel matrix $\boldsymbol{K}_e$, the image owner decrypts $\boldsymbol{K}_e$ to recover $\boldsymbol{K}$, by using *Ordinary Decryption* or *Batch Decryption*.

- *Ordinary Decryption.* $\boldsymbol{K}_e$ is decrypted as $\boldsymbol{K}[i,j] \leftarrow VHE.Dec(\boldsymbol{K}_e[i,j], \boldsymbol{S})$ element-by-element, where $i = 1, \ldots, n$ and $j = 1, \ldots, n$. Obviously, the time complexity is $\mathcal{O}(n^2)$, which brings heavy computational burden to the image owner.
- *Batch Decryption.* $\boldsymbol{K}_e$ is decrypted in a batch manner based on packaging $\boldsymbol{S}$ as illustrated in Eq. (24).

$$\boldsymbol{S}_{pack} = \begin{bmatrix} \boldsymbol{S} & 0 & 0 & \cdots & 0 \\ 0 & \boldsymbol{S} & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & \boldsymbol{S} \end{bmatrix} \cdot \begin{bmatrix} k_1 \\ k_2 \\ \cdots \\ k_{n^2} \end{bmatrix}. \tag{24}$$

For each element of $\boldsymbol{K}_e$, the corresponding keys are the same, i.e., $\boldsymbol{S}$. Accordingly, all ciphertexts can be packed into one ciphertext, and thus only a single decryption operation is required to recover $\boldsymbol{K}$, which significantly decreases computation costs for the image owner.

The $\boldsymbol{K}[i,j]$ of $\boldsymbol{K}$ is the inner product between HOG^i and HOG^j. We denote HOG^i as H_i. Hence, the kernel matrix $\boldsymbol{K}$ is as follows:

$$\boldsymbol{K} = \begin{bmatrix} H_1 \cdot H_1 & \cdots & H_1 \cdot H_n \\ H_2 \cdot H_1 & \cdots & H_2 \cdot H_n \\ \cdots & \cdots & \cdots \\ H_n \cdot H_1 & \cdots & H_n \cdot H_n \end{bmatrix}. \tag{25}$$

D. Secure SVM Trainer

Secure SVM training model can be constructed from the kernel matrix $\boldsymbol{K}$ as follows.

1) Secure Kernel Functions Calculation: We can securely calculate three kinds of kernel functions, namely, the linear, polynomial, and Gaussian kernels.

Linear Kernel: The linear kernel is calculated as

$$\boldsymbol{K}_{ij} = H_i^T H_j. \tag{26}$$

Thus, $\boldsymbol{K}_{ij}$ is only acquired from the element in $\boldsymbol{K}$ according to the index i and j.

Polynomial Kernel: The polynomial kernel is calculated as

$$\boldsymbol{K}_{ij} = (H_i^T H_j + \gamma)^d. \tag{27}$$

The polynomial kernel can be obtained based on linear kernel by adding a constant γ and then doing d power operation.

Gaussian Kernel: The Gaussian kernel is calculated as

$$\boldsymbol{K}_{ij} = e^{-\frac{\|H_i - H_j\|^2}{2\sigma^2}}. \tag{28}$$

As $\|H_i - H_j\|^2 = H_i^T H_i + H_j^T H_j - 2H_i^T H_j$, we have

$$\boldsymbol{K}_{ij} = e^{-\frac{\boldsymbol{K}_{ii} + \boldsymbol{K}_{jj} - 2\boldsymbol{K}_{ij}}{2\sigma^2}}. \tag{29}$$

Since the cloud cannot extract the real HOG descriptor $H_i, i = \{1, 2, \ldots, n\}$ from the above computations, the security of kernel functions can be guaranteed.

Algorithm 3: Secure SMO.

Input: $\boldsymbol{K}$, $\boldsymbol{y} = \{y_1, y_2, \ldots, y_n\}$
Output: $\boldsymbol{\alpha}, b$

initialize $\boldsymbol{\alpha}$ and b
choose two samples (H_i, y_i) and (H_j, y_j)
calculate
$E_i = \sum_{k=1}^{n} \boldsymbol{\alpha}_k y_k \boldsymbol{K}_{ik}, E_j = \sum_{k=1}^{n} \boldsymbol{\alpha}_k y_k \boldsymbol{K}_{jk}$
calculate $\boldsymbol{\alpha}_j^{new} = \alpha_j^{old} + \frac{y_j(E_i - E_j)}{\boldsymbol{K}_{ii} + \boldsymbol{K}_{jj} - 2\boldsymbol{K}_{ij}}$
calculate

$$U = \begin{cases} \max\{0, \boldsymbol{\alpha}_j^{old} - \boldsymbol{\alpha}_i^{old}\}, & y_i y_j = -1 \\ \max\{0, \boldsymbol{\alpha}_j^{old} + \boldsymbol{\alpha}_i^{old} + C\}, & y_i y_j = 1 \end{cases}$$

and

$$V = \begin{cases} \min\{C, \boldsymbol{\alpha}_j^{old} - \boldsymbol{\alpha}_i^{old} + C\}, & y_i y_j = -1 \\ \min\{0, \boldsymbol{\alpha}_j^{old} + \boldsymbol{\alpha}_i^{old}\}, & y_i y_j = 1 \end{cases}$$

if $\boldsymbol{\alpha}_j^{new} > V$ **then**
  $\boldsymbol{\alpha}_j^{new} = V$
else if $\boldsymbol{\alpha}_j^{new} < U$ **then**
  $\boldsymbol{\alpha}_j^{new} = U$
end if
calculate $\boldsymbol{\alpha}_i^{new} = \boldsymbol{\alpha}_i^{old} + y_i y_j (\boldsymbol{\alpha}_j^{old} - \boldsymbol{\alpha}_j^{new})$
calculate $b_i = -E_i - y_i \boldsymbol{K}_{ii} (\boldsymbol{\alpha}_i^{new} - \boldsymbol{\alpha}_i^{old}) - y_j \boldsymbol{K}_{ij} (\boldsymbol{\alpha}_j^{new} - \boldsymbol{\alpha}_j^{old}) + b)$ and $b_j = -E_j - y_i \boldsymbol{K}_{ij} (\boldsymbol{\alpha}_i^{new} - \boldsymbol{\alpha}_i^{old}) - y_j \boldsymbol{K}_{jj} (\boldsymbol{\alpha}_j^{new} - \boldsymbol{\alpha}_j^{old}) + b)$
if $0 < \boldsymbol{\alpha}_i^{new} < C$ **then**
  $b^{new} = b_i$
else if $0 < \boldsymbol{\alpha}_j^{new} < C$ **then**
  $b^{new} = b_j$
else
  $b^{new} = \frac{b_i + b_j}{2}$
end if
if all samples satisfy KKT condition, stop training.

2) Secure SMO Algorithm: Given the kernel matrix $\boldsymbol{K}$ and the corresponding labels $\boldsymbol{y} = \{y_1, y_2, \ldots, y_n\}$, where $y_i \in \{+1, -1\}$, the secure SMO algorithm is designed, which is depicted in Algorithm 3.

E. Correctness of Kernel Function Calculation

In the original SMO algorithm [43], only the training data are used to calculate kernel functions. Here, Alg 3 is roughly the same with the original one except that some operations that are concentrated on computing kernel functions are performed over the ciphertexts for the training data. Hence, as long as the kernel functions can be correctly performed over ciphertexts, Alg 3 can obtain the same training result as the original SMO algorithm does. Furthermore, in Alg 3, the kernel functions are calculated based on the homomorphic operations of inner product. As the homomorphism of VHE can ensure the correctness of kernel function calculations over encrypted training data, the trained SVM model in the ciphertext domain will be correct.

VI. Security Analysis

In this section, we discuss the security of PPPD in two aspects, the security of VHE and the privacy of the kernel matrix.

A. Security of VHE

The security of VHE depends on the learning with error (LWE) problem, that is, given polynomially many samples of $(\boldsymbol{a}_i, b_i) \in \mathbb{Z}_q^m \times \mathbb{Z}_q$ satisfying

$$b_i = \boldsymbol{v}^T \boldsymbol{a}_i + \epsilon_i, \tag{30}$$

where the error term $\epsilon \in \mathbb{Z}_q$ is generated from the Gaussian probability distribution χ, no adversary can have a non-negligible advantage to acquire the vector $\boldsymbol{v} \in \mathbb{Z}_q^m$.

Theorem 1: If the LWE problem is hard, the advantage to retrieve the new secret key matrix $\boldsymbol{S}_{new}$ from $\boldsymbol{S}_{new}\boldsymbol{M} = \boldsymbol{S}_{old}^* + \boldsymbol{E}$ is negligible, where $\boldsymbol{M}$, $\boldsymbol{S}_{old}$ and $\boldsymbol{E}$ are the key-switching, the old secret key and the error matrices respectively, and $(\cdot)^*$ is the binary representation of a matrix.

The proof details are referred to [19]. Simply speaking, if LWE is hard, it is hard to solve $\boldsymbol{S}_{new}\boldsymbol{M} = \boldsymbol{S}_{old}^* + \boldsymbol{E}$. Hence, it is impossible to retrieve the new secret key $\boldsymbol{S}_{new}$ on the premise of the known $\boldsymbol{M}$. Therefore, no adversary can recover the original images.

B. Privacy of Kernel Matrix

Since the labels $y_i \in \{+1, -1\}$ will not leak the sensitive information, even if they are not encrypted, the data privacy would be reduced to the security of the kernel matrix $\boldsymbol{K} \in \mathbb{Z}^{n \times n}$, where n is the number of training samples. That is, even if the cloud obtains $\boldsymbol{K}$, it cannot acquire any elements from the HOG feature matrix $\boldsymbol{X} \in \mathbb{Z}^{m \times n}$ containing n HOG feature vectors $\boldsymbol{x}_i \in \mathbb{Z}^m, i = 1, \ldots, n$. This can be ensured due to the following reasons.

Firstly, a pair of standard orthogonal matrices can be constructed as

$$\boldsymbol{I} = \boldsymbol{Q}^T \boldsymbol{Q}, \boldsymbol{Q} \in \mathbb{R}^{n \times n},$$

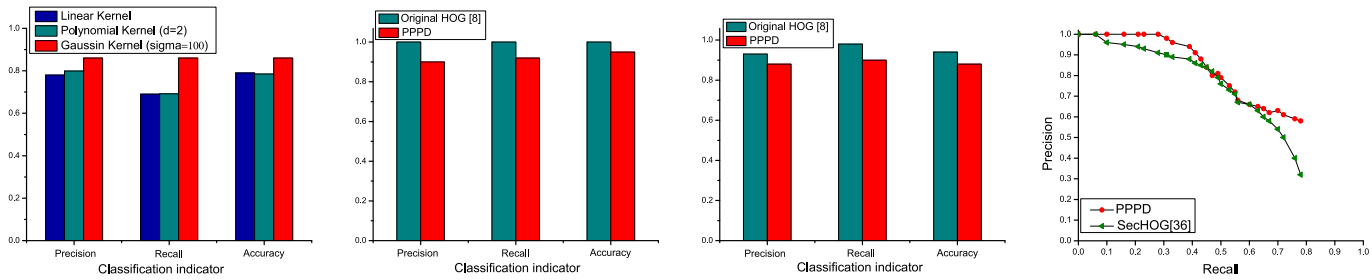
where $\boldsymbol{I}$ is an identity matrix of $n \times n$. The column vector of $\boldsymbol{Q}$ constitutes a set of standard orthogonal bases in $\mathbb{R}^n$. Since there is an infinite number of standard orthogonal bases in $\mathbb{R}^n$, there is an infinite number of matrices $\boldsymbol{Q}$ satisfying $\boldsymbol{I} = \boldsymbol{Q}^T \boldsymbol{Q}$. Therefore, given $\boldsymbol{K} = \boldsymbol{X}^T \boldsymbol{X}$, we have

$$\begin{aligned} \boldsymbol{K} &= \boldsymbol{X}^T \boldsymbol{X} = \boldsymbol{X}^T \boldsymbol{I} \boldsymbol{X} \\ &= \boldsymbol{X}^T \boldsymbol{Q}^T \boldsymbol{Q} \boldsymbol{X} \\ &= (\boldsymbol{Q}\boldsymbol{X})^T (\boldsymbol{Q}\boldsymbol{X}). \end{aligned}$$

Let $\boldsymbol{A} = (\boldsymbol{Q}\boldsymbol{X})$. As there is an infinite number of $\boldsymbol{Q}$, there is an infinite number of $\boldsymbol{A}$. Further, there is an infinite number of $\boldsymbol{X}$ satisfying $\boldsymbol{X}^T \boldsymbol{X} = \boldsymbol{K}$. The cloud, thus, cannot determine $\boldsymbol{X}$ only by the kernel matrix $\boldsymbol{K}$. As a result, the privacy of the kernel matrix $\boldsymbol{K}$ can be guaranteed. Also, since the SMO training algorithm is designed based on $\boldsymbol{K}$, its security can be achieved if $\boldsymbol{K}$ is secure.

TABLE III
CLASSIFICATION PERFORMANCE ON MULTIPLE DATASETS

	Attribute		Precision		Recall		Accuracy	
Dataset	**Train**	**Test**	**Original**	**Our**	**Original**	**Our**	**Original**	**Our**
RGB-D camera [45]	5000	2200	1	1	1	0.94	1	0.97
Daimler Pedestrian Detection [46]	22404	3996	1	1	0.76	0.68	0.88	0.84
USC Pedestrian Detection [47]	507	200	1	0.99	1	0.98	1	0.98
Caltech Pedestrian Detection [48]	3525	3683	0.51	0.49	0.63	0.68	0.51	0.48
PennFudanPed [49]	260	80	1	1	1	0.93	1	0.96



(a) (b) (c) (d)

Fig. 6. Classification performance. (a) Multiple kernel functions (b) Comparison with original HOG on MIT (c) Comparison with original HOG on INRIA (d) Comparison with SecHOG.

VII. PERFORMANCE EVALUATION

In this section, the performance of PPPD is evaluated. The experiments are conducted on a server with E5-2430 CPU and 32 GB RAM, running Ubuntu 16.04, and a laptop with i5-4130 CPU and 8 GB RAM, running Windows 10. In addition, our simulations are carried out on multiple pedestrian datasets, including MIT [44] and INRIA [8].

A. Classification Performance

The classification results of three kernel functions are firstly compared to select a proper one to conduct the experiments. Then, the accuracy between the original HOG [8] and PPPD is compared over MIT dataset [44] and INRIA dataset [8], respectively. Besides, in Table III, the experiments for evaluating the performance of PPPD are executed on multiple pedestrian image datasets. Finally, PPPD is compared with SecHOG [36] in terms of classification precision. Three frequently used indicators, including precision, recall, and accuracy, are measured for the classification performance. Note that for the classification performance indicators, $Precision = \frac{TP}{TP+FP}$, $Recall = \frac{TP}{TP+FN}$ and $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$, where TP is *True Positive*, TN is *True Negative*, FP is *False Positive* and FN is *False Negative* [50].

1) Multiple Kernel Functions: In Fig. 6(a), three kernel functions are compared on the INRIA dataset [8] in terms of three indicators. PPPD performs better with the Gaussian kernel than others. Therefore, the Gaussian kernel would be chosen for subsequent experiments. Note that in Fig. 6(a), the *sigma* is the precision parameter of the Gaussian kernel function. The d represents the polynomial degree in the polynomial kernel function.

2) Comparison With Original HOG: Here, we compare PPPD with the original plaintext algorithm of HOG on MIT dataset [44] and INRIA dataset [8], respectively. Firstly, MIT [44] contains 924 pedestrian images, 80% and 20% of which are used for training and prediction, respectively. Since MIT [44] does not have any negative sample, we select images with the same scenes (no pedestrians) as negative samples. In Fig. 6(b), PPPD and the original HOG have similar classification performance. The difference in detection accuracy is at most 4%. For INRIA [8], the training set has 614 positive samples and 1218 negative samples; the test set has 288 positive samples (including 1126 pedestrians) and 453 negative samples. In Fig. 6(c), PPPD still has the approximate performance to the original HOG, though the INRIA dataset [8] has more complex pedestrian backgrounds. Consequently, PPPD not only achieves the gain of privacy protection but also preserves the good classification performance.

3) Multiple Pedestrian Datasets: To illustrate the wide availability of PPPD in various pedestrian detection scenarios, multiple pedestrian datasets are used, including RGB-D [45], Daimler [46], USC [47], Caltech [48] and PennFudanPed [49], which contain pedestrian images in different poses and scenes. In Table III, over multiple pedestrian image datasets, PPPD and the original scheme have close performance, but PPPD performs on the ciphertexts of images with privacy preservation.

4) Comparison With SecHOG: PPPD is compared with SecHOG [36] over the INRIA dataset. In Fig. 6(d), the classification effects of PPPD approximate that of SecHOG. Specifically, when the recall rate exceeds 70%, PPPD has a slight advantage

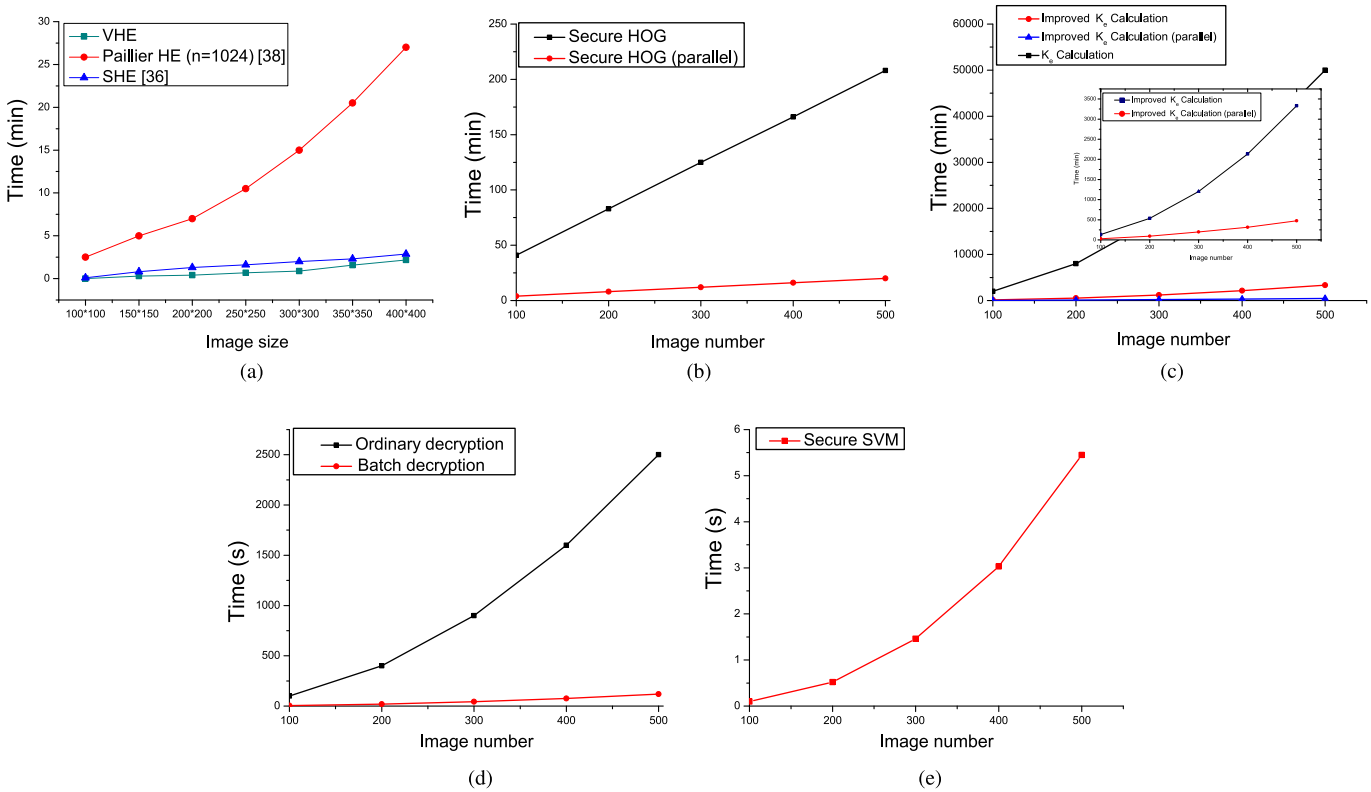


Fig. 7. Time costs. (a) Time cost on image encryption (b) Time cost on HOG extraction (c) Time cost on $\boldsymbol{K}_e$ calculation (d) Time cost on $\boldsymbol{K}_e$ decryption (e) Time cost on secure SVM training.

TABLE IV
TIME COST ON INITIALIZATION PHASE

Generated key and matrix	Time (s)
$\boldsymbol{S}$	0.001
$\boldsymbol{M}_L/\boldsymbol{M}_R$	1.25
$\boldsymbol{M}_H$	0.15
$\boldsymbol{M}_0/\boldsymbol{M}_{45}/\boldsymbol{M}_{90}/\boldsymbol{M}_{135}$	0.08
$\boldsymbol{M}_{up}/\boldsymbol{M}_{down}$	1.01

in precision. Due to VHE, PPPD has lower computational and communication costs than SecHOG.

B. Computation Cost

Time consumption of each phase in PPPD is collected when it processes 500 images simultaneously that are randomly selected in the MIT dataset [44].

1) Initialization: In this phase, the image owner generates keys and key-switching matrices corresponding to linear-transforming operations. In the Table IV, the time for matrix generation is no more than 2 seconds. These matrices are calculated once in initialization.

2) Image Encryption: Image encryption should be lightweight due to the limited computing capability of image owners. We compare VHE with Paillier HE [38] and SHE [36] in the time of image encryption. As shown in Fig. 7(a), VHE takes much less time than Paillier HE. For example, for the image size of 400×400, the encryption time of VHE is less than 3 minutes but that of Paillier HE is up to 27 minutes. Although VHE in PPPD and SHE in SecHOG have close performance in terms of computational cost, due to the usage of VHE in PPPD, the encrypted kernel matrix enables the generation of the secure SVM trainer from the extracted encrypted HOG features. It is not necessary to decrypt and re-encrypt the HOG features for the image owners. Therefore, compared with SecHOG, PPPD can achieve high accuracy of pedestrian detection with lower computational overhead.

3) Secure HOG Extraction: After the image owner uploads encrypted images, the cloud sequentially calculates secure HOG feature extraction. In Fig. 7(b), the extraction time of HOG features increases linearly with the number of images. When the number reaches 500, the time to extract features from 500 images is considerable, i.e., 220 minutes. However, the time cost can be significantly reduced by using parallel computing; that is, only 20 minutes are needed for feature extraction of 500 images.

4) Ciphertext Kernel Matrix Calculation: After extracting encrypted HOG features, inner products of features are calculated in the ciphertext domain, and the kernel matrix is constructed. For n encrypted HOG features with k block descriptors, the time complexities of ciphertext kernel matrix calculation are $\mathcal{O}(n^2\ k)$ in Algorithm 1 and $\mathcal{O}(n^2)$ in Algorithm 2, respectively. In Fig. 7(c), although the Algorithm 2 saves much time with the increasing number of images, the time cost for the cloud server is high, e.g., 100 images would cost 180 minutes. To

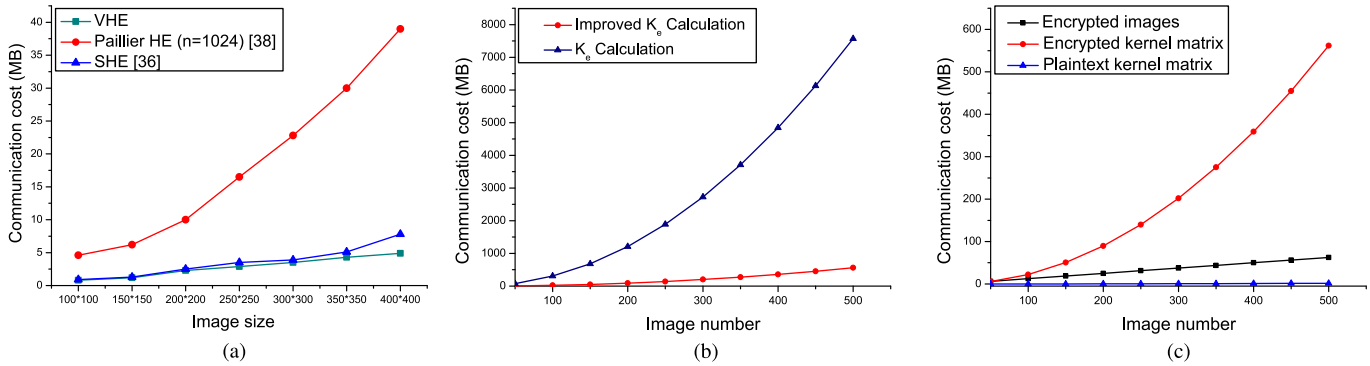


Fig. 8. Communication costs. (a) Communication cost on encrypted images (b) Communication cost on $\boldsymbol{K}_e$ (c) Overall communication costs.

reduce the time cost of $\boldsymbol{K}_e$ generation, the multi-process parallel computing mode can be utilized. In Fig. 7(c), it takes 28 minutes to complete the same tasks. As the number of images increases, the computing time would be reduced to a larger extent.

5) Ciphertext Kernel Matrix Decryption: In this phase, $\boldsymbol{K}_e$ is returned to the image owner for decryption. In Fig. 7(d), the time complexity of the method of ordinary decryption is $\mathcal{O}(n^2)$, and for $\boldsymbol{K}_e$ containing 500 images, the decryption time is up to 2500 seconds. However, for batch decryption, decrypting the same $\boldsymbol{K}_e$ only costs 112 seconds.

6) Secure SVM Trainer: The cloud performs secure SVM training after obtaining $\boldsymbol{K}$. In Fig. 7(e), because inner products can be calculated in advance, it takes only about 6 seconds to complete the training of 500 images.

C. Communication Overhead

We discuss the communication overhead between the image owner and the cloud.

1) Encrypted Images: We compare VHE with Paillier HE [38] and SHE [36] in the size of encrypted images. As shown in Fig. 8(a), VHE and SHE have close performance in terms of communication cost, consuming much less bandwidth than Paillier HE. For example, for the image size of 400×400, the communication cost of VHE is less than 5 MB but that of Paillier HE is up to 40 MB.

2) Ciphertext Kernel Matrix: A comparison of communication overhead for transmitting $\boldsymbol{K}_e$ is presented. For n HOG features with k block descriptors, space complexities are $\mathcal{O}(n^2\ k)$ and $\mathcal{O}(n^2)$ according to Algorithm 1 and Algorithm 2, respectively. As shown in Fig. 8(b), the improved Alg.2 greatly reduces communication cost. For example, for $\boldsymbol{K}_e$ containing 500 images, it consumes the bandwidth of 562 MB, while the communication cost of the original Algorithm 1 is up to 7568 MB.

3) Overall Communication Cost: For the whole process of feature extraction and training, the image owner needs to interact with the cloud to upload encrypted images, download $\boldsymbol{K}_e$, and upload $\boldsymbol{K}$. In Fig. 8(c), communication costs of uploading $\boldsymbol{D}_e$ and $\boldsymbol{K}$ are quite low, and that of downloading $\boldsymbol{K}_e$ is practically acceptable.

TABLE V
COMMUNICATION COST COMPARISON

Communication Cost (MB)	PPPD	SCPD
Encrypted original images	24.41	12750
Encrypted kernel or encrypted features	562	23400
Plaintext kernel or re-encrypted features	1.67	410.15
Total	588.08	36560.15

D. Comparison of Communication Costs

We compare PPPD with the straightforward approach by simply combining the secure HOG [36] and the secure SVM [40] in terms of the communication cost, which is called SCPD. In SCPD, the PEGASOS and FSCL algorithms [40] are used in SVM and the secure HOG is under single-server model [36]. we use 500 images with the size of 128×64 from MIT database [44], and the corresponding HOG features are the vectors with 1680 dimensions. The encryption schemes, including BCP [41], VHE [19] and SHE [39], are involved in the comparison of communication costs. We firstly set the security level $k = 80$ and specify the ciphertext size-related parameters for BCP [41], VHE [19] and SHE [39]. In BCP, $\log N = 1024$ and the ciphertext size is 512 Bytes with the modulus N^2. In VHE, $l = 100, q \approx 2^{50}$, $w = 2^{20}$, and the ciphertext size is 400 Bytes for a 64 dimensional vector. In SHE, $p = 2$ for the plaintext space of $\mathbb{F}_p[x] = \Phi_n(x)/(f(x))$, and $\Phi_n(x)$ is factored modulo 2 into multiple irreducible factors to support the single instruction multiple data (SIMD). According to the test results in [51], the fresh ciphertext size is about 204 KB for a 64 dimensional vector, and the evaluated ciphertext size is at least 46.8 MB for a 1680 dimensional vector.

Table V demonstrates the comparisons about the communication costs between PPPD and SCPD. The communication costs for 500 encrypted images are 24.41 MB and 12 750 MB in PPPD and SCPD, since the ciphertext size of each row (i.e., a 64 dimensional vector) in one image are 400 Bytes and 204 KB in VHE and SHE, respectively. To download encrypted kernel matrix and upload decrypted kernel matrix for 500 images, PPPD consumes the bandwidths of 565 MB and 1.67 MB, respectively, as illustrated in Fig. 8(c). However, in SCPD, the communication costs for 500 images are 23 400 MB and

TABLE VI
COMPARISON OF EXISTING SCHEMES

Scheme	Security Technique	Security of Raw Image	Security of HOG Feature	Security of SVM Training	Communication Cost
Original HOG [8]	×	×	×	×	low
PPHOG [35]	VHE	√	√	×	low
SecHOG [36]	SHE + Multi-Server	√	√	×	high
Secure SVM [40]	BCP	×	√	√	high
SCPD	SHE+BCP	√	√	√	high
PPPD	VHE	√	√	√	low

410.15 MB, as one 1680 dimensional HOG feature encrypted by SHE [39] and BCP [41] has the ciphertext sizes of 46.8 MB and 512 Bytes, respectively. Therefore, PPPD is more efficient than SCPD in terms of the communication cost. In PPPD, by taking advantages of VHE [19], the image can be batch encrypted in row-by-row, rather than pixel-by-pixel like BCP [41]. Although SHE [39] can leverage SIMD to pack multiple pixels into one vector for batched encryption, the ciphertext size dramatically increases with the vector dimensions (e.g., 204 KB with 64 dimensions but 46.8 MB with 1680 dimensions). Besides, in PPPD, the encrypted kernel matrix enables the generation of the secure SVM trainer from the extracted encrypted HOG features. It is not necessary to decrypt and re-encrypt the HOG features for the image owners compared with SCPD.

E. Comparison of Existing Schemes

We have summarized and compared the detailed features of the existing schemes and our proposed PPPD, which are shown in TABLE VI. The symbols of "√" represents that the scheme achieves that property, and "×" represents that the scheme does not achieve that property. The original HOG scheme [8] has low communication cost, but it does not adopt any security techniques, thus it cannot provide any privacy protection. Although PPHOG [35] and SecHOG [36] can guarantee the privacy of raw images and HOG features, can guarantee the privacy of raw images and HOG features, they cannot preserve the privacy for SVM training. Besides, due to the multi-server technique, SecHOG brings high communication overheads. Secure SVM [40] can preserve the privacy of SVM training. Nevertheless, no privacy protection mechanism is designed for extracted HOG features. SCPD can protect the privacy for raw images, extracted HOG features, and SVM training. but it does not preserve the privacy of exacted HOG features. SCPD can protect the privacy for raw images, extracted HOG features, and SVM training, but the transmission of encrypted HOG features between the cloud server and the image owner would cause large communication costs. Our PPPD achieves the privacy protection of raw images, HOG features and SVM training with low communication costs.

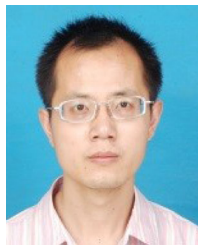
VIII. CONCLUSION

In this paper, we have proposed an accurate image-based pedestrian detection scheme (PPPD) over the encrypted pedestrian images with privacy preservation. The private linear-transforming matrices are designed to generate the SVM trainer from the secure HOG extractors to avoid the retrieval of the encrypted extractors for image owners in model training, such that the communication overhead can be dramatically reduced. The construction of linear-transforming matrices has the independence of value, potentially being applied to many other privacy-preserving image classification algorithms. PPPD achieves high classification accuracy over multiple pedestrian datasets with low computational and communication costs, exhibiting its applicability in autonomous driving. For the future work, we will investigate the privacy-preserving face identification over surveillance videos that provides high accuracy and efficiency guarantees.

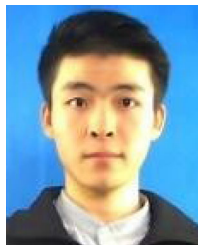
REFERENCES

[1] Y. Han, Q. Li, F. Wang, B. Wang, K. Mizuno, and Q. Zhou, "Analysis of pedestrian kinematics and ground impact in traffic accidents using video records," *Int. J. Crashworthiness*, vol. 24, no. 2, pp. 211–220, 2019.
[2] 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.
[3] L. Dong, D. Sun, G. Han, X. Li, Q. Hu, and L. Shu, "Velocity-free localization of autonomous driverless vehicles in underground intelligent mines," *IEEE Trans. Veh. Technol.*, vol. 69, no. 9, pp. 9292–9303, Sep. 2020.
[4] K. Ren, Q. Wang, C. Wang, Z. Qin, and X. Lin, "The security of autonomous driving: Threats, defenses, and future directions," *Proc. IEEE*, vol. 108, no. 2, pp. 357–372, Feb. 2020.
[5] T. Zhang, S. Liu, C. Xu, and H. Lu, "Mining semantic context information for intelligent video surveillance of traffic scenes," *IEEE Trans. Ind. Informat.*, vol. 9, no. 1, pp. 149–160, Feb. 2013.
[6] P.-H. Chiu, P.-H. Tseng, and K.-T. Feng, "Interactive mobile augmented reality system for image and hand motion tracking," *IEEE Trans. Veh. Technol.*, vol. 67, no. 10, pp. 9995–10 009, Oct. 2018.
[7] A. Mateus, D. Ribeiro, P. Miraldo, and J. C. Nascimento, "Efficient and robust pedestrian detection using deep learning for human-aware navigation," *Robot. Auton. Syst.*, vol. 113, pp. 23–37, 2019.
[8] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2005, pp. 886–893.
[9] S. Zhang, H. Li, Y. Dai, J. Li, M. He, and R. Lu, "Verifiable outsourcing computation for matrix multiplication with improved efficiency and applicability," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 5076–5088, Aug. 2018.
[10] M. Villari, M. Fazio, S. Dustdar, O. Rana, and R. Ranjan, "Osmotic computing: A new paradigm for edge/cloud integration," *IEEE Cloud Comput.*, vol. 3, no. 6, pp. 76–83, Nov. 2016.
[11] C. E. Lawson, "Pixels, porn, and private selves: Intimacy and authenticity in the celebrity nude photo hack," *Celebrity Stud.*, vol. 6, no. 4, pp. 607–609, 2015.
[12] J. Daemen and V. Rijmen, *The Design of Rijndael: AES-The Advanced Encryption Standard*. Berlin, Germany: Springer Science & Business Media, 2013.
[13] Y. Zhang, Y. Lu, H. Nagahara, and R.-I. Taniguchi, "Anonymous camera for privacy protection," in *Proc. IEEE Int. Conf. Pattern Recognit.*, 2014, pp. 4170–4175.
[14] S.-Z. Chen, C.-C. Guo, and J.-H. Lai, "Deep ranking for person re-identification via joint representation learning," *IEEE Trans. Image Process.*, vol. 25, no. 5, pp. 2353–2367, May 2016.

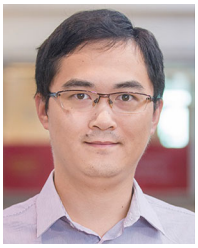
[15] N. Miyazaki, K. Tsuji, M. Zheng, M. Nakashima, Y. Matsuda, and E. Segawa, "Privacy-conscious human detection using low-resolution video," in *Proc. IEEE Asian Conf. Pattern Recognit.*, 2015, pp. 326–330.
[16] Y. Wang *et al.*, "Resource aware person re-identification across multiple resolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 8042–8051.
[17] G. Brazil, X. Yin, and X. Liu, "Illuminating pedestrians via simultaneous detection & segmentation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 4950–4959.
[18] D. Xu, W. Ouyang, E. Ricci, X. Wang, and N. Sebe, "Learning cross-modal deep representations for robust pedestrian detection," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5363–5371.
[19] H. Zhou and G. Wornell, "Efficient homomorphic encryption on integer vectors and its applications," in *Proc. IEEE Inf. Theory Appl. Workshop*, 2014, pp. 1–9.
[20] D. Liu, A. Alahmadi, J. Ni, X. Lin, and X. Shen, "Anonymous reputation system for IIoT-enabled retail marketing atop PoS blockchain," *IEEE Trans. Ind. Informat.*, vol. 15, no. 6, pp. 3527–3537, Jun. 2019.
[21] M. Abadi *et al.*, "Deep learning with differential privacy," in *Proc. ACM Conf. Comput. Commun. Secur.*, ACM, Oct. 2016, pp. 308–318.
[22] J. Ni, K. Zhang, K. Alharbi, X. Lin, N. Zhang, and X. Shen, "Differentially private smart metering with fault tolerance and range-based filtering," *IEEE Trans. Smart Grid*, vol. 8, no. 5, pp. 2483–2493, Sep. 2017.
[23] J. Ding, Y. Huang, W. Liu, and K. Huang, "Severely blurred object tracking by learning deep image representations," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 2, pp. 319–331, Feb. 2016.
[24] J. Li, X. Liang, S. Shen, T. Xu, J. Feng, and S. Yan, "Scale-aware fast R-CNN for pedestrian detection," *IEEE Trans. Multimedia*, vol. 20, no. 4, pp. 985–996, Apr. 2018.
[25] K. Lin and M. Chen, "On the design and analysis of the privacy-preserving SVM classifier," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 11, pp. 1704–1717, Nov. 2011.
[26] Y. Chen, X. Zhu, and S. Gong, "Person re-identification by deep learning multi-scale representations," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2590–2600.
[27] P. Mohassel and P. Rindal, "Aby 3: A mixed protocol framework for machine learning," in *Proc. ACM Conf. Comput. Commun. Secur.*, Oct. 2018, pp. 35–52.
[28] E. Makri, D. Rotaru, N. P. Smart, and F. Vercauteren, "Epic: Efficient private image classification," in *Proc. CT-RSA*, 2019, pp. 473–492.
[29] K. Yuan *et al.*, "Stealthy porn: Understanding real-world adversarial images for illicit online promotion," in *Proc. IEEE Symp. Secur. Privacy*, 2019, pp. 952–966.
[30] K. Eykholt *et al.*, "Robust physical-world attacks on deep learning visual classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 1625–1634.
[31] G. Xu, H. Li, S. Liu, M. Wen, and R. Lu, "Efficient and privacy-preserving truth discovery in mobile crowd sensing systems," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3854–3865, Apr. 2019.
[32] C. Huang, R. Lu, J. Ni, and X. Shen, "Dapa: A decentralized, accountable, and privacy-preserving architecture for car sharing services," *IEEE Trans. Veh. Technol.*, vol. 69, no. 5, pp. 4869-4882, May 2020.
[33] J. Liang, Z. Qin, S. Xiao, L. Ou, and X. Lin, "Efficient and secure decision tree classification for cloud-assisted online diagnosis services," *IEEE Trans. Dependable Secure Comput.*, to be published, doi: 10.1109/TDSC.2019.2922958.
[34] Y. Zhang, C. Xu, H. Li, K. Yang, J. Zhou, and X. Lin, "HealthDep: An efficient and secure deduplication scheme for cloud-assisted ehealth systems," *IEEE Trans. Ind. Informat.*, vol. 14, no. 9, pp. 4101–4112, Sep. 2018.
[35] H. Yang, Y. Huang, Y. Yu, M. Yao, and X. Zhang, "Privacy-preserving extraction of hog features based on integer vector homomorphic encryption," in *Proc. Inf. Secur. Pract. Experience*, 2017, pp. 102–117.
[36] Q. Wang, J. Wang, S. Hu, Q. Zou, and K. Ren, "SecHOG: Privacy-preserving outsourcing computation of histogram of oriented gradients in the cloud," in *Proc. ACM ASIA Conf. Comput. Commun. Secur.*, 2016, pp. 257–268.
[37] L. Jiang, C. Xu, X. Wang, B. Luo, and H. Wang, "Secure outsourcing sift: Efficient and privacy-preserving image feature extraction in the encrypted domain," *IEEE Trans. Dependable Secure Comput.*, vol. 17, no. 1, pp. 179–193, Jan./Feb. 2020.
[38] P. Paillier and D. Pointcheval, "Efficient public-key cryptosystems provably secure against active adversaries," in *Proc. ASIACRYPT Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, 1999, pp. 165–179.
[39] Z. Brakerski and V. Vaikuntanathan, "Fully homomorphic encryption from ring-lwe and security for key dependent messages," in *Proc. Adv. Cryptol. CRYPTO*, 2011, pp. 505–524.
[40] F.-J. Gonzalez-Serrano, A. Navia-Vazquez, and A. Amor-Martin, "Training support vector machines with privacy-protected data," *Pattern Recognit.*, vol. 72, pp. 93–107, Dec. 2017.
[41] E. Bresson, D. Catalano, and D. Pointcheval, "A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications," in *Proc. Adv. Cryptol., ASIACRYPT*, 2003, pp. 37–54.
[42] P. Weinzaepfel, H. Jegou, and P. Perez, "Reconstructing an image from its local descriptors," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2011, pp. 337–344.
[43] J. Platt, "Sequential minimal optimization: A fast algorithm for training support vector machines," in *Proc. Adv. Kernel Methods: Support Vector Learn.*. Cambridge, MA, USA: MIT Press, 1998, pp. 212–223.
[44] C. Papageorgiou and T. Poggio, "A trainable system for object detection," *Int. J. Comput. Vis.*, vol. 38, no. 1, pp. 15–33, 2000.
[45] H. Liu, J. Luo, P. Wu, S. Xie, and H. Li, "People detection and tracking using RGB-D cameras for mobile robots," *Int. J. Adv. Robot. Syst.*, vol. 13, no. 5, pp. 1–11, 2016.
[46] F. Flohr and D. Gavrila *et al.*, "Pedcut: An iterative framework for pedestrian segmentation combining shape models and multiple data cues," in *Proc. Brit. Mach. Vis. Conf.*, 2013, pp. 1–11.
[47] B. Wu and R. Nevatia, "Detection of multiple, partially occluded humans in a single image by Bayesian combination of edgelet part detectors," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2005, pp. 90–97.
[48] P. Dollar, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: An evaluation of the state of the art," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 4, pp. 743–761, Apr. 2012.
[49] L. Wang, J. Shi, G. Song, and I.-F. Shen, "Object detection combining recognition and segmentation," in *Proc. Asian Conf. Comput. Vis.*, 2007, pp. 189–199.
[50] C. Sammut and G. I. Webb, *Encyclopedia of Machine Learning*. Berlin, Germany: Springer Science & Business Media, 2011.
[51] M. Varia, S. Yakoubov, and Y. Yang, "HetEst: A homomorphic encryption testing framework," in *Proc. Financial Cryptogr.*, 2015, pp. 213–230.



Haomiao Yang (Member, IEEE) received the M.S. and Ph.D. degrees in computer applied technology from the University of Electronic Science and Technology of China (UESTC), China, in 2004 and 2008, respectively. He is currently an Associate Professor with the School of Computer Science and Engineering, UESTC. From 2012 to 2013, he worked as a Postdoctoral Fellow with Kyungil University. From 2018 to 2019, he also worked as a Visiting Scholar with the University of Waterloo. His research interests include cryptography, cloud security, and cyber security for aviation communication.



Qixian Zhou received the B.S. degree in biomedical engineering from Southwest Medical University, Lu Zhou, China. He is currently working toward the M.S. degree in computer science with the University of Electronic Science and Technology of China. His current research interests include data security and artificial intelligence security.



Jianbing Ni (Member, IEEE) received the Ph.D. degree in electrical and computer engineering from the University of Waterloo, Waterloo, Canada, in 2018. He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, Queen's University, Kingston, Canada. His research interests include applied cryptography and network security, with current focus on edge computing, mobile crowdsensing, Internet of Things, and Blockchain technology.

Hongwei Li (Senior Member, IEEE) received the Ph.D. degree from the University of Electronic Science and Technology of China (UESTC), China, in 2008. He is currently the Department Head and a Professor with the Department of Information Security, School of Computer Science and Engineering, UESTC. From 2011 to 2012, he worked as a Postdoctoral Fellow with the University of Waterloo. He has authored or coauthored more than 80 technical papers. His research interests include network security and applied cryptography. He is the Associate Editor for the IEEE INTERNET OF THINGS JOURNAL, and *Peer-to-Peer Networking and Applications*, the Guest Editor for the IEEE NETWORK and the IEEE INTERNET OF THINGS JOURNAL. He was the recipient of the Best Paper Award from the IEEE MASS 2018 and the IEEE HELTHCOM 2015. He is the Distinguished Lecturer of IEEE Vehicular Technology Society.

Xuemin (Sherman) Shen (Fellow, IEEE) received the Ph.D. degree in electrical engineering from Rutgers University, New Brunswick, NJ, USA, in 1990. He is currently a University Professor with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. His research interests include resource management in interconnected wireless/wired networks, wireless network security, social networks, smart grid, and vehicular ad hoc and sensor networks. He is a Registered Professional Engineer with Ontario, Canada, an Engineering Institute of Canada Fellow, a Canadian Academy of Engineering Fellow, a Royal Society of Canada Fellow, and a Distinguished Lecturer with the IEEE Vehicular Technology Society and Communications Society.

He was the recipient of the R.A. Fessenden Award in 2019 from IEEE, Canada, the James Evans Avant Garde Award in 2018 from the IEEE Vehicular Technology Society, the Joseph LoCicero Award in 2015, and the Education Award in 2017 from the IEEE Communications Society. He has also received the Excellent Graduate Supervision Award in 2006 and the Outstanding Performance Award 5 times from the University of Waterloo and the Premier's Research Excellence Award (PREA) in 2003 from the Province of Ontario, Canada. He was the Technical Program Committee Chair/Co-Chair for the IEEE Globecom'16, the IEEE Infocom'14, the IEEE VTC'10 Fall, the IEEE Globecom'07, the Symposia Chair for the IEEE ICC'10, the Tutorial Chair for the IEEE VTC'11 Spring, the Chair for the IEEE Communications Society Technical Committee on Wireless Communications, and P2P Communications and Networking. He is the President of the IEEE Communications Society.