4-2022

SecureAD: A secure video anomaly detection framework on convolutional neural network in edge computing environment

Hang CHENG
Fuzhou University

Ximeng LIU
Singapore Management University, xmliu@smu.edu.sg

Huaxiong WANG
Nanyang Technological University

Yan FANG
Fujian Agricultural University

Meiqing WANG
Fuzhou University

See next page for additional authors

Citation

CHENG, Hang; LIU, Ximeng; WANG, Huaxiong; FANG, Yan; WANG, Meiqing; and ZHAO, Xiaopeng. SecureAD: A secure video anomaly detection framework on convolutional neural network in edge computing environment. (2022). *IEEE Transactions on Cloud Computing*. 107, (2), 1413-1427.
Available at: https://ink.library.smu.edu.sg/sis_research/5932

Published in IEEE Transactions on Cloud Computing, 2020, Advance online. DOI: 10.1109/TCC.2020.2990946

SecureAD: A Secure Video Anomaly Detection Framework on Convolutional Neural Network in Edge Computing Environment

Hang Cheng, Ximeng Liu, Huaxiong Wang, Yan Fang, Meiqing Wang, and Xiaopeng Zhao

Abstract—Anomaly detection offers a powerful approach to identifying unusual activities and uncommon behaviors in real-world video scenes. At present, convolutional neural networks (CNN) have been widely used to tackle anomalous events detection, which mainly rely on its stronger ability of feature representation than traditional hand-crafted features. However, massive video data and high cost of CNN model training are a challenge to achieve satisfactory detection results for resource-limited users. In this paper, we propose a secure video anomaly detection framework (SecureAD) based on CNN. Specifically, we introduce additive secret sharing to design several calculation protocols for achieving safe CNN training and video anomaly detection. Besides, we propose a Bloom filter based fine-grained access control policy to authenticate legitimate users, without leaking the privacy of raw personal attributes. In addition, edge computing instead of cloud computing is integrated into the architecture to reduce response time between servers and users in an outsourced environment. Finally, we prove that the proposed SecureAD achieves secure video anomaly detection without compromising the privacy of the related data. Also, the simulation results demonstrate the effectiveness and security of our SecureAD.

Index Terms—Privacy-preserving, anomaly detection, Bloom filter, CNN, secret sharing.

1 Introduction

In recent years, surveillance cameras are widely applied in security field, such as crime forensics, traffic analysis, and infant care. With the rapid growth of surveillance cameras, it was estimated that more than 560 petabytes per day are generated from global surveillance cameras [1]. Due to dual properties of both time and space from video data, rich spatiotemporal information has attracted extensive attention. As a fundamental challenge over video data, the detection of abnormal events in video streams has aroused great interests from both academia and industry in recent years [2], [3]. Video anomaly detection aims to automatically alarm to anomalous events, such as unexpected falls, illegal entry, fighting while determining the time window of being an anomaly in a long video sequence. To this end, most traditional methods focus on how to design hand-crafted feature representations for better filtering anomaly from regular events [4], [5]. However, the hand-crafted features are generally extracted from low-level appearance and motion cues, making them challenging to obtain superior detection performance.

Recently, deep learning technology with convolutional neural networks (CNN) has enjoyed immense popularity due to its strong capability of feature learning. At present, deep learning technology has been introduced to tackle the task of video anomaly detection. In particular, autoencoder (AE) networks that have done well in object tracking and face alignment are used to capture spatiotemporal cue of abnormal events [2]. Besides, the long short term memory (LSTM), known as an essential tool of analyzing the temporal events, was successfully exploited to address the problem of the spatial motion anomaly from the temporal view [6]. Despite their successes that are little achieved by the hand-crafted features, the substantial computational costs and high hardware configurations are not affordable for resource-limited users during the complicated CNN's model training. Furthermore, the vast volumes of video data also pose a significant challenge to general users. Directly employing cloud computing technique is a universal solution to solve the above issue. However, its complexly centralized management suffers from significant bandwidth consumption. Alternatively, edge computing can offer the benefit to allow the storage/computation services to be performed at the local area network [7]. Thus, data integrity and timely response can be ensured. Nevertheless, it will raise concerns on data privacy due to the loss of data control for users. The same issue exists in cloud computing. To solve this issue, some homomorphic encryption (HE) based attempts have been made gradually for CNN [8], [9], [10]. High computational costs and data extension, however, are a big barrier for HE in a real-world application.

In this paper, we propose a secure, lightweight video anomaly detection framework (SecureAD), which integrates AE and LSTM techniques in convolution manner. With

- *H. Cheng is with the College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350108, China; the School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore. E-mail: hcheng@fzu.edu.cn.*
- *X. Liu and M. Wang are with the College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350108, China. X. Liu is also with Key Lab of Information Security of Network Systems (Fuzhou University), Fujian Province, China. E-mail: snbnix@gmail.com, mqwang@fzu.edu.cn.*
- *H. Wang is with the School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore. Email: hxwang@ntu.edu.sg.*
- *Y. Fang is with the College of Computer and Information Sciences, Fujian Agriculture and Forestry University, Fuzhou 350002, China. Email: yfang@fafu.edu.cn.*
- *X. Zhao is with the School of Computer Science and Software Engineering, Dept. of Cryptography and Cyber Security at East China Normal University, China. Email: 52164500025@stu.ecnu.edu.cn.*

SecureAD, the edge servers can train the CNN model over encrypted video streams, and provide secure anomaly detection service. Specifically, we develop several additive secret sharing based calculation protocols to achieve privacy-preserving CNN training, where no raw video data and CNN model parameters are revealed. And also, a novel access control mechanism is designed to realize outsourced user authentication, where users' attributes are kept confidential to servers. Furthermore, edge computing is introduced into SecureAD to reduce communication costs and response latency, thus making our framework practical. To summarize, the main contributions of SecureAD are as follows:

- To the best of our knowledge, this work is the first attempt to address the privacy-preserving CNN-based anomaly detection task over outsourced surveillance videos. The SecureAD allows the edge servers to detect video anomaly, without compromising the privacy of the video data and model parameters.
- We design a series of protocols of performing SecureAD using additive secret sharing technique. With them, the computational costs and communications overheads are much lower than HE. Besides, no keys are involved in all types of computation operations related to SecureAD.
- We build a Bloom filter based access control mechanism to allow the servers to check the validity of the users' identities, without leaking users' real attributes.
- We employ edge computing to design a novel architecture of video anomaly detection. In this architecture, users and servers are combined together to carry out video anomaly detection in a non-interactive manner. Meanwhile, it can improve the response latency, which cannot be solved by traditional cloud computing.

In the rest of the paper, we first present some preliminaries in Section 2. Then, we introduce the problem formulations including system model, problem statement, security model, and design goals in Section 3. In addition, some subprotocols are presented in Section 4. Furthermore, the detail of the proposed SecureAD is shown in Section 5. Finally, the analysis of the correctness, security, and performance are given in Section 6. Some related work is reviewed in Section 7. Conclusions are drawn in Section 8.

2 PRELIMINARIES

In this section, we briefly recall the necessary background on CNN, Bloom filters, and additive secret sharing on which our SecureAD is based.

2.1 Convolutional Autoencoder (CAE)

As a variation of autoencoder, CAE is generally composed of convolution layer, pooling layer, deconvolution layer, and unpooling layer. The former two layer types appear in the encoding stage, and others belong to the decoding stage. More details are given in Supplemental Materials A.1.

2.2 Convolutional Long Short Term Memory (CLSTM)

Long short term memory (LSTM) is the most commonly used neural sequence model, which successfully prevents backpropagated errors from gradients vanishing/exploding by introducing the concepts of forget gate, input gate, and output gate. Nevertheless, the architecture based on the full connection makes LSTM hard to learn the spatial correlation information from inputs. As a variant of LSTM, CLSTM is firstly proposed by Shi et al. in [11] to solve the precipitation nowcasting issue. Different from LSTM, the convolution operations instead of matrix operations are performed to calculate the feature maps, thus largely reducing the number of the model training parameters. Supplemental Materials A.2 describes some fundamental equations associated with the forget/input/ouput gates that are indispensable parts of CLSTM.

2.3 Bloom Filter (BF)

In general, a BF is considered as a binary vector $\mathcal{B}$ of m bits, $\mathcal{B} = \{b_1, \ldots, b_m\}$, in which the independent hash functions are configured to compute the element values in $\mathcal{B}$. The core idea of BF is to employ these hash functions $h_i(x)$ ($i \in [1, k]$) to map each element x of a set to an element of $\{1, \ldots, m\}$, and then set the value at the corresponding position in $\mathcal{B}$ as one. Given a new element y, BF technique can determine that it belongs to the $\mathcal{B}$ by the equation $\sum_{i=1}^{k} b_{h_i(y)} = k$.

Although BF has no false negatives, a low-probability false positive could still happen. As described in Eq. 1, the probability of false positives is influenced by the three factors, namely, the hash function number k, and the BF size m, and the original element number n. Through regulating these factors, the probability of false positives can be controlled within an accepted range for practical applications.

$$Pr(BF) = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^{k} \approx \left(1 - e^{-kn/m}\right)^{k}. \quad (1)$$

2.4 Additive Secret Sharing protocols

Additive secret sharing technique allows each participant to obtain the share of the sum of some secrets by locally adding its received shares of all secrets, namely, having additive homomorphic property. In SecureAD, this useful property is introduced to develop some privacy-preserving computation protocols, which serve as the basis of the privacy-preserving CNN model training. It is worth noting that our SecureAD is used in an environment that a pair of parties $(\mathcal{P}_1, \mathcal{P}_2)$ is involved in calculating a joint function of their private inputs. To facilitate further discussion, two atomic additive secret sharing protocols are briefly described below.

- **Secure Addition Protocol (SecAdd).** The goal of this protocol is to securely compute the sum of the private numbers u and v, namely $f(u, v) = u + v$. It is straightforward for the $\mathcal{P}_1$ and $\mathcal{P}_2$ to achieve without the interaction according to the homomorphic property of the additive secret sharing.
- **Secure Multiplication Protocol (SecMul).** The **SecMul** takes as input the private numbers u and v, and outputs f_1 and f_2, following $f = f_1 + f_2 = u \cdot v$. Compared to the **SecAdd**, this protocol based on *Beaver's*

triplet [12] is relatively complex, which additionally involves the random number provider $\mathcal{R}$ and the interaction between participants $\mathcal{P}_i$ ($i \in \{1, 2\}$).

More details of the above two protocols are given in Supplemental Materials B.1 and B.2.

3 Problem Formulation

To better understand, some notations related to the problem formulation are given in TABLE 1.

TABLE 1
Notation descriptions for problem formulation.

Notations	Descriptions
CO	Content owner
ES	Edge server
RP	Random number provider
AU	Authorized user
CNN	Convolutional neural network
$\mathcal{A}^*_{i,j}$	Adversary targeting parties i, j

3.1 System Model

As shown in Fig. 1, our framework mainly includes Content Owner (CO), Edge Server (ES), Random Number Provider (RP), and Authorized User (AU).

- The CO splits each video frame and access control policy into the two random shares, and then separately sends them to two different ESs for storage.
- The ESs take charge of the CNN model training while providing the anomaly detection for AUs. Also, the ESs can verify the access effectiveness of a user.
- The RP is responsible for generating some random numbers for the secret sharing protocols.
- The AU is authorized by CO in our SecureAD. An AU is able to decrypt and identify actual anomalous events from the returned alarmed video frame shares. Certainly, if AU has sufficient permissions, it can also be allowed to upload its video shares to the two ESs for obtaining the detection results.

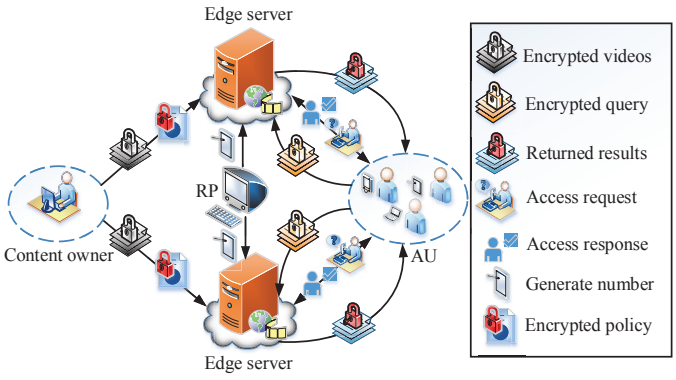


Fig. 1. The infrastructure of SecureAD.

3.2 Problem Statement

Consider massive surveillance videos that may include anomalous events. Video data should be encrypted before uploading to ESs for CNN model training or anomaly detection. An AU can be allowed to identify the final anomaly detection results or upload the encrypted videos to the ESs for detecting anomalous events. When receiving the detection request, ESs can verify the access legitimacy of AUs. Due to that all outsourced video data are encrypted during the model training and anomaly detection, our proposed framework needs to address the following issues.

- Since homomorphic encryption leads to high computation costs and data extension, we need some efficient, lightweight cryptographic protocols to perform complex linear or nonlinear operations for CNN, without compromising the users' privacy.
- In order to free the key management in our framework, a keyless cipher primitive should be designed to encrypt the outsourced video data and all intermediate results.
- To support outsourced access control, a secure, efficient access authentication mechanism needs to be built for the servers to check the validity of users' identification, such that the servers do not learn about user's attribute information (e.g., name, address, and telephone number) during the access authentication.

3.3 Security Model

The edge servers involved in our SecureAD are assumed to follow the *honest-but-curious* model, which is widely used in the signal processing in the encrypted domain [13], [14], [15], [16]. Under this security model, ES honestly performs the designated protocol specification, but curiosity makes it prone to analyze the private information, such as video contents and access policy, in its storage and convolutional neural network parameters during CNN model training. Besides, we assume that two edge servers do not collude with each other. In our scheme, RP just generates a set of random numbers. Therefore, we assume that CO directly chooses an honest client from its company as RP.

Under the assumption of the *honest-but-curious* model, three types of active adversaries, $\mathcal{A}^*_{1,j}$, $\mathcal{A}^*_{2,j}$, $\mathcal{A}^*_{3,j}$ ($j \in \{1, 2\}$) are considered in our attack model, which is typically adopted in [17]. These types separately challenge CO's share, AU's share, and RP's share, which will be sent to the j-th ES. Specifically, $\mathcal{A}^*_{i,j}$ is assumed to possess the following capabilities. First, it is able to monitor the party i (refers to CO or AU or RP) and the ES j communications, and get the corresponding share data. Second, it may compromise the ES j to have access to all the encrypted data stored at the ES j. In this case, it also obtains some encrypted data sent from another edge server by invoking an interactive protocol. Third, it may perform statistical analysis of previous anomaly detection requests, and then infer some specific information, such as determine whether or not the current user has logged in. Note that either the same or different types of adversaries cannot cooperate with two ESs simultaneously in our SecureAD.

3.4 Design Goals

To enable privacy-preserving event anomaly detection for outsourced surveillance videos under the above security model, SecureAD should achieve the following goals:

- *Correctness*. To guarantee the excellent performance of video anomaly detection in the encrypted domain, our SecureAD should be designed to have the ability of correctly training CNN model over encrypted videos, and obtain almost the same detection performance as the plaintext counterpart.
- *Data Privacy*. SecureAD should prevent the edge servers from learning the video data and detected results. Also, the privacy of the CNN model parameters and users' attributes should be protected during the implementation of our SecureAD.
- *Efficiency*. To enable resource-limited users to enjoy the quality service of anomaly detection, our SecureAD should be able to provide low computation/communication costs for users.
- *Outsourced Access Authentication*. Generally, a trusted access control center is dedicated to manage and authenticate users' attributes, which decreases the portability of our SecureAD. For decentralization, the outsourcing of secure access authentication should be supported in SecurAD, in which the privacy of users' attributes is not leaked to ESs.

4 Secret Sharing Based Secure Computation Protocols

In this Section, we design a series of new powerful sub-protocols based on **SecAdd** and **SecMul**, which can be properly combined to securely realize some complex computational functions emerged in a convolutional neural network. To explain more clearly, we assume each edge server $\mathcal{P}_i$ ($i \in \{1,2\}$) holds a pair of (u_i, v_i) sent by CO or AUs, in which u_i and v_i are additive shares of the two secrets u and v, respectively, satisfying $u = u_1 + u_2$ and $v = v_1 + v_2$. A common goal of the following protocols is that two $\mathcal{P}_i$ corporately compute $f(u,v)$ and separately output f_1 and f_2, s.t., $f = f_1 + f_2$, where two $\mathcal{P}_i$ do not compromise the individual inputs and outputs with each other. Some important notations are listed in TABLE 2.

TABLE 2
Notation descriptions during the sub-protocol construction.

Notations	Descriptions
$\mathcal{R}$	Random number provider
$\mathcal{P}_i$ ($i \in \{1,2\}$)	The i-th edge server
u_i ($i \in \{1,2\}$)	The i-th share of private number u ($u = u_1 + u_2$)
v_i ($i \in \{1,2\}$)	The i-th share of private number v ($v = v_1 + v_2$)
f_i ($i \in \{1,2\}$)	The i-th output of protocol f ($f = f_1 + f_2$)
SecAdd/SecMul	Secure addition/multiplication protocol
SecXor/SecCom	Secure Xor/comparison protocol
SecMax/SecCon	Secure maximum/convolutional protocol
SecRec/SecExp	Secure reciprocal/natural exponential protocol

4.1 Secure XOR Protocol (SecXor)

Let u and v be two private bits. Observe that $u \oplus v = u + v - 2u \cdot v$ for any $u, v \in \{0,1\}$. It means that XOR operation "$\oplus$" in $\mathbb{Z}_2$ is a linear combination of addition and multiplication over a finite field $\mathbb{F}$. Therefore, we can achieve secure XOR operation between u and v by invoking **SecAdd** and **SecMul**. Specifically, two $\mathcal{P}_i$ with private shares u_i and v_i ($i \in \{1,2\}$) jointly compute $(g_1, g_2) = \textbf{SecMul}(u_1, u_2, v_1, v_2)$, where $u = u_1 + u_2$ and $v = v_1 + v_2$. Then, $\mathcal{P}_i$ computes and outputs $f_i = \textbf{SecAdd}(\textbf{SecAdd}(u_i, v_i), -2g_i)$ without interaction. As a result, $u \oplus v = f_1 + f_2$ can be securely obtained over $\mathbb{F}$. To simplify expression, $\textbf{SecXor}(u, v)$ instead of $\textbf{SecXor}(u_1, u_2, v_1, v_2)$ will be used in the subsequent discussions. Other protocols also refer to this expression.

4.2 Secure Comparison Protocol (SecCom)

Although the literature [15] introduces a secure solution to the comparison of two private numbers $u, v \in \mathbb{R}$, it is suitable only to the field of $\mathbb{Z}_2$ and is not compatible with other additive secret sharing protocols in $\mathbb{F}$. In this paper, we improve it and propose a practical secure comparison protocol **SecCom** over $\mathbb{F}$. Specifically, we transform the comparison problem into the identification of the sign of $u - v$ (denoted as d). For a signed integer, the sign is indicated by its most significant bit (MSB). For a non-integer x, it can be encoded as an integer $\bar{x}$ through scaling by the factor 10^κ, $\bar{x} = x \cdot 10^\kappa$, where κ is the length of the fractional part of x. Thus we focus on secure MSB extraction of $d \cdot 10^\kappa$ in our **SecCom** as follows.

First, $\mathcal{R}$ generates ϱ random bits $r^{(0)}, \ldots, r^{(\varrho-1)}$ and computes $r = -r^{(\varrho-1)} \cdot 2^{\varrho-1} + \sum_{k=0}^{\varrho-2} r^{(k)} \cdot 2^k$ based on two's complement encoding that represents signed numbers in binary way. Then $\mathcal{R}$ splits r into two random shares a_1 and a_2, where $r = a_1 + a_2$. Similarly, $\mathcal{R}$ generates shares $r^{(k)} = r_1^{(k)} + r_2^{(k)}, k \in [0, \varrho - 1]$. Then, $\mathcal{R}$ sends the share a_i and $r_i^{(k)}$ to $\mathcal{P}_i$. Note that this step can be done offline and $a_i \neq -r_i^{(\varrho-1)} \cdot 2^{\varrho-1} + \sum_{k=0}^{\varrho-2} r_i^{(k)} \cdot 2^k, i \in \{1,2\}$. Second, each $\mathcal{P}_i$ computes locally its share $d_i = u_i - v_i$ of difference $d = u - v$ using **SecAdd**, and converts d_i into integer $\lfloor d_i \cdot 10^\kappa \rfloor$, which is conveniently denoted by d_i. Following up, $\mathcal{P}_i$ computes $d_i - a_i$ and reveals it to another ES. In this case, $\mathcal{P}_i$ has a public number $s = d - r$ and private shares $r_i^{(k)} (k \in [0, \varrho - 1])$. Third, ESs collaboratively compute the sum d of s and r just using bit-based XOR "$\oplus$" and AND "$\wedge$" operations, where $d^{(k)} = s^{(k)} \oplus r^{(k)} \oplus t^{(k)}$ and $t^{(k+1)} = (s^{(k)} \wedge r^{(k)}) \oplus ((s^{(k)} \oplus r^{(k)}) \wedge t^{(k)})$. It is obvious that the XOR and AND operations over signed integers can be securely achieved by invoking the **SecXor** and **SecMul** protocols in $\mathbb{F}$, whereas the comparison method in [15] is subject to $\mathbb{Z}_2$. The expected result is $d^{(\varrho-1)} = d_1^{(\varrho-1)} + d_2^{(\varrho-1)}$, namely, the MSB of number $u - v$. If $d^{(\varrho-1)} = 0$, $u \geq v$, else otherwise. More details are described in Algorithm 1.

4.3 Secure Maximum Protocol (SecMax)

Given n private numbers $\{u_1, \ldots, u_n\}$, the purpose of **SecMax** protocol is to ensure that each edge server can find the maximum from these numbers. During the execution of this protocol, n numbers including the maximum need keep unknown to any edge server. In order to maintain privacy, we generalize two numbers-based **SecCom** protocol to find the maximum of n ($n > 2$) numbers. A naive approach to find the maximum is to repeatedly compare the current

Algorithm 1: Secure Comparison between Numbers

Input: $\mathcal{P}_i$ has shares u_i and v_i $(u_i, v_i \in \mathbb{F}, i \in \{1,2\})$.
Output: $\mathcal{P}_i$ outputs $d_i^{(\varrho-1)} (i \in \{1,2\})$.

$\mathcal{R}$ generates ϱ random bits $r^{(0)}, \ldots, r^{(\varrho-1)}$ and computes $-r^{(\varrho-1)} \cdot 2^{\varrho-1} + \sum_{k=0}^{\varrho-2} r^{(k)} \cdot 2^k \to r$;
$\mathcal{R}$ randomly splits $r^{(k)} \to r_1^{(k)} + r_2^{(k)}$ and $r \to a_1 + a_2$;
$\mathcal{R}$ sends $r_i^{(k)}$ and a_i to $\mathcal{P}_i$, i$\in \{1,2\}$;
$\mathcal{P}_i$ calculates $\lfloor (u_i - v_i) \cdot 10^{\kappa} \rfloor \to d_i$;
$\mathcal{P}_i$ calculates $d_i - a_i \to s_i$ and sends another ES;
$\mathcal{P}_1$ and $\mathcal{P}_2$ set $0 \to t_i^{(0)}$, and calculate **SecXor**$(s^{(0)}, r^{(0)}) \to (d_1^{(0)}, d_2^{(0)})$;
for *every bit position,* $k = 1$ *to* $\varrho - 1$ **do**
  $\mathcal{P}_1$ and $\mathcal{P}_2$ jointly calculate:
  SecXor$(s^{(k)}, r^{(k)}) \to (\alpha_1^{(k)}, \alpha_2^{(k)})$;
  SecMul$(s^{(k)}, r^{(k)}) \to (\beta_1^{(k)}, \beta_2^{(k)})$;
  SecMul$(\alpha^{(k)}, t^{(k-1)}) \to (\alpha_1^{(k)}, \alpha_2^{(k)})$;
  SecXor$(\beta^{(k)}, \alpha^{(k)}) \to (d_1^{(k)}, d_2^{(k)})$;
end
$\mathcal{P}_i$ returns $d_i^{(\varrho-1)}$.

maximum against the next adjacent number by using **SecCom** as described before. Obviously, this approach needs to be $n-1$ rounds to achieve the maximum, namely, the time complexity is $O(n)$.

To improve the search speed for the maximum, **SecMax** takes the paradigm of binary search trees as reference. However, the difference is that we construct the binary tree T from the bottom upward instead of top-down manner. In **SecMax**, **SecCom** protocol is typically performed on a pair of adjacent numbers at each level of T. Due to the independence of pairs, integrating the parallel technique into T can reduce the time complexity of searching for the maximum from $O(n)$ to $O(\log_2 n)$. After construction, the root node $\alpha_1^{(h)}$ of T is taken as the maximum of n numbers. The detailed construction of **SecMax** is shown in Algorithm 2 in Supplemental Materials B.3.

4.4 Secure Reciprocal Protocol/Natural Exponential Protocol (SecRec/SecExp)

Here, our **SecRec** and **SecExp** protocols are modified versions of the *SecInv* and *SecExp* in [14], respectively. To be specific, we use the proposed **SecCom** to replace the counterparts in the *SecInv* and *SecExp*. The details of the **SecRec** and **SecExp** protocols are shown in Supplemental Materials B.4 and B.5.

4.5 Secure Convolutional Protocol (SecCon)

The details of **SecCon** are in Supplemental Materials B.6.

5 Privacy-preserving video anomaly detection fusing CAE and CLSTM

In this section, we propose a privacy-preserving video anomaly detection framework, as shown in Fig. 2. In this framework, both CAE and CLSTM are assembled in a unified convolutional network, in which this combination is often adopted to capture the sufficient spatiotemporal information in computer vision issues [6]. For ease of expression, hereafter, the superscript signs "$\tilde{}$" and "$\hat{}$" are used to denote the shares, which are distributed to two edge servers.

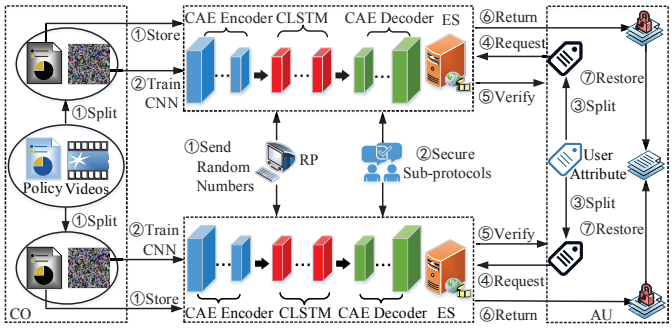


Fig. 2. Framework of SecureAD scheme.

5.1 Secure Outsourced Access Control Mechanism

To realize fine-grained access control, the attribute-based public-key cryptosystem with ciphertext-policy (CP-ABE) introduced by Bethencourt [18] may be an appropriate approach to determine which users can enjoy the anomaly detection service. However, the key escrow will raise the privacy concerns on users' private keys. Therefore, we propose a Bloom filter based keyless outsourced access control mechanism, which allows ESs to perform user authentication while keeping the access control policy and users' attributes confidential to the ESs.

Assume that our SecureAD contains n attributes $\{A_1, A_2, \ldots, A_n\}$. Let $\mathcal{V}_i = \{v_{i,1}, v_{i,2}, \ldots, v_{i,n_i}\}$ be a set of possible values of A_i $(i \in [1,n])$. After a new CO registers the system, the CO first needs to define an access policy $P = \{P_1, P_2, \ldots, P_n\}$ $(P_i \subset \mathcal{V}_i)$ that only allows the users with the matching attributes to perform video anomaly detection on its pre-trained CNN model. Following, the CO employs the Bloom filter technique to generate a Bloom filter $\mathcal{B}_i = \{b_{i,1}, b_{i,2}, \ldots, b_{i,m_i}\}$ associated with the access policy $P_i = \{p_{i,1}, p_{i,2}, \ldots, p_{i,\ell_i}\}$ $(i \in [1,n])$, where a set of the hash function $\mathcal{H}_i = \{h_{i,1}, h_{i,2}, \ldots, h_{i,k_i}\}$ is used to compute P_i. More specifically, for each element $p_{i,s}$ of P_i, CO calculates $h_{i,t}(p_{i,s})$ and sets $b_{h_{i,t}(p_{i,s})}$ as 1, where $s \in [1, \ell_i]$ and $t \in [1, k_i]$. Thus, CO can obtain a set of Bloom filters $\{\mathcal{B}_i\}$ $(i \in [1,n])$ corresponding to its defined access policy P. Next, CO splits each Bloom filter $\mathcal{B}_i$ $(i \in [1,n])$ into the two random shares, and then sends them to the corresponding edge server.

Let user's attribute value list be $\mathcal{V}_{user} = \{v_{1,l_1}, v_{2,l_2}, \ldots, v_{n,l_n}\}$, where $v_{i,l_i} \in \mathcal{V}_i$. Similarly, $\mathcal{V}_{user}$ is transformed into n Bloom filters $\mathcal{Q}_i = \{q_{i,1}, q_{i,2}, \ldots, q_{i,m_i}\}$ $(i \in [1,n])$, each of which has the same bit-length as $\mathcal{B}_i$. And also, $\mathcal{Q}_i$ is divided randomly into two shares, and sent to edge servers $\mathcal{P}_1$ and $\mathcal{P}_2$, respectively. When obtaining the user authentication request, two ESs corporately calculate $b_{i,j} \wedge q_{i,j}$ for any $i \in [1,n]$ and $j \in [1,m_i]$ by directly invoking **SecMul**, namely, $(s_{i,j}, t_{i,j}) =$ **SecMul**$(b_{i,j}, q_{i,j})$, where $b_{i,j} \wedge q_{i,j} = s_{i,j} + t_{i,j}$. And then, $\mathcal{P}_1$ and $\mathcal{P}_2$ independently computes $\sum_{j=1}^{m_i} s_{i,j}$, $\sum_{j=1}^{m_i} t_{i,j}$, and sends them to each other. This step implies that each $\mathcal{P}_i$ can know

$a_i = \sum_{j=1}^{m_i} s_{i,j} + \sum_{j=1}^{m_i} t_{i,j}$, which is equal to the sum of element-wise AND of $\mathcal{B}_i$ and $\mathcal{Q}_i$ ($i \in [1, n]$). According to the knowledge in Subsection 2.3, if all $a_i (i \in [1, n])$ are equal to k, each ES can identify that the attribute value list of the current user matches with CO's defined access policy P. Otherwise, the user is determined to be illegal.

Remark: There are few possible values for some specific attributes in real applications, such as gender. In this case, merging these attributes into one attribute is a wise solution to reduce computation complexity.

5.2 Secure CNN fusing CAE and CLSTM

In the following, we will use those above elementary secure protocols to construct some sophisticated security protocols to achieve secure CAE and CLSTM.

5.2.1 Secure CAE

As described in Subsection 2.1, CAE generally includes the convolution/deconvolution layer and pooling/unpooling layer. Here, we will give the detail of these layers during privacy-preserving forward/backward propagation.

A. Secure Forward Propagation

Convolution Layer. Based on Eq.15 in Supplemental Materials A.1, the value of each neuron in any convolution layer can be calculated by adopting the convolution and activation operations. As for the convolution operation, it can be securely done through performing the proposed **SecCon** protocol. Thus, how to securely carry out the activation operation become desired. In CAE of our framework, we take the sigmoid function $\sigma(x) = 1/(1 + e^{-x})$ as activation operation. It is very apparent that the sigmoid function consists of three basic operations, natural exponential, addition, and reciprocal. Therefore, we can combine **SecExp**, **SecAdd**, and **SecRec** protocols to achieve the secure computation of the sigmoid function by the following two steps.

Step-1: The server $\mathcal{P}_i$ runs the **SecExp** protocol to output f_i, s.t., $f_1 + f_2 = e^{-x}$.

Step-2: The server $\mathcal{P}_1$ calculates $g_1 = 1 + f_1$ and $\mathcal{P}_2$ sets $g_2 = f_2$, where $g = g_1 + g_2 = 1 + e^{-x}$. Thus, servers $\mathcal{P}_1$ and $\mathcal{P}_2$ jointly calculate $(h_1, h_2) =$**SecRec**(g) by using the **SecRec** protocol.

As a result, the sigmoid function with the input x, $\sigma(x) = 1/(1 + e^{-x})$ can be securely obtained by directly calculating the summation of h_1 and h_2. To simplify the representation, we denote the secure computation protocol of $\sigma(x)$ as **SecSig**. With the **SecSig** protocol, the edge servers $\mathcal{P}_1$ and $\mathcal{P}_2$ jointly calculate the value of each neuron in the convolution layer, without learning about any useful information. More specifically, once obtaining secure version of Eq. 15 in Supplemental Materials A.1, $\mathcal{P}_1$ and $\mathcal{P}_2$ calculate

$$(\widetilde{a}_{i,j}^{(m,l)}, \widehat{a}_{i,j}^{(m,l)}) = \textbf{SecSig}\left(\sum_{k=1}^{K_l} W^{(k,m,l)} * A^{(k,l-1)} + b^{(m,l)}\right), \tag{2}$$

where $a_{i,j}^{(m,l)} = \widetilde{a}_{i,j}^{(m,l)} + \widehat{a}_{i,j}^{(m,l)}$, and the secure versions of "*" and "+" operations can be achieved by invoking the **SecCon** and **SecAdd** protocols.

Pooling Layer. Using the **SecAdd** protocol, we readily calculate the average-pooling in the encrypted domain. Let the value of neuron at position (i, j) in the m^{th} channel in the l^{th} layer as $\lambda_{i,j}^{(m,l)}$, which can be calculated by the following equation.

$$\lambda_{i,j}^{(m,l)} = \frac{1}{n \times n}\left(\sum_{s=1}^{n}\sum_{t=1}^{n} a_{i+s,j+t}^{(m,l-1)}\right), \tag{3}$$

where the size of pooling filter is set to $n \times n$. Based on Eq. 3, $\mathcal{P}_i (i \in \{1, 2\})$ individually computes the desired share of $\lambda_{i,j}^{(m,l)}$ by repeatedly calling the protocol **SecAdd**.

Deconvolution Layer. Deconvolution layer is a reverse operation of the convolution layer. Essentially, it works the same way as the convolution layer. Therefore, the above convolution layer can be taken as a reference to obtain a secure version of the deconvolution layer.

Unpooling Layer. The unpooling layer carries out the upsampling operation, which reverses the pooling operation. In this layer, the edge servers $\mathcal{P}_1$ and $\mathcal{P}_2$ only need to set the value of neuron at position ($i + s$, $j + t$) in the m^{th} channel in the l^{th} layer as

$$(\widetilde{\lambda}_{i,j}^{(m,l+1)}, \widehat{\lambda}_{i,j}^{(m,l+1)}) \rightarrow (\widetilde{\lambda}_{i+s,j+t}^{(m,l)}, \widehat{\lambda}_{i+s,j+t}^{(m,l)}),$$

where let the l^{th} layer be unpooling layer. And $s, t \in [1, n]$.

B. Secure Backward Propagation

Backward propagation usually adopts a gradient descent method to update the weights, where the different neurons are set to different weights according to those neuron errors that are propagated from the total loss. In the following, our focus is to achieve secure weight updating process.

Assume that the $\delta_i^{(l)}$ is the error matrix of the i^{th} feature map in the l^{th} layer. If the l^{th} layer is the pooling layer and the next layer is the convolution layer, $\delta_i^{(l)}$ is securely calculated by the edge servers through the following equation.

$$(\widetilde{\delta}_i^{(l)}, \widehat{\delta}_i^{(l)}) = \left(\sum_{j=1}^{M} \textbf{SecCon}(\delta_j^{(l+1)}, w_{i,j}^{(l+1)})\right),$$

where $w_{i,j}^{(l+1)}$ denotes the weight matrix of the i^{th} feature map in the j^{th} kernel connected to the $(l+1)^{th}$ layer. M is the kernel number in the $(l+1)^{th}$ layer. If the l^{th} layer is the convolution layer and the $(l+1)^{th}$ layer is the pooling layer, we just perform the upsampling operation on $\delta_j^{(l+1)}$.

Based on the error matrix $\delta_i^{(l)}$, we can deduce the following updating equations of weight matrixes by using proposed secure protocols.

$$(\widetilde{W}_{new}, \widehat{W}_{new}) = \textbf{SecAdd}\left(W_{old}, -\eta \cdot \nabla W\right),$$
$$(\nabla\widetilde{W}, \nabla\widehat{W}) = \textbf{SecCon}(A_i^{(l-1)}, \delta_j^{(l)}),$$

where "∇" denotes gradient, and η indicates the learning rate that is public to all participants. $A_i^{(l-1)}$ is the i^{th} feature map in the $(l-1)^{th}$ layer.

Meanwhile, we can derive the updating equations of biases by

$$(\widetilde{b}_{new}, \widehat{b}_{new}) = \textbf{SecAdd}\left(b_{old}, -\eta \cdot \nabla b\right), (\nabla\widetilde{b}, \nabla\widehat{b}) = \sum_{u,v} \delta_j^{(l)},$$

where (u, v) denotes the neuron coordinate of the j feature map in the l^{th} layer.

5.2.2 Secure CLSTM

As introduced in Subsection 2.2, CLSTM is mainly composed of forget gate, input gate, and output gate. The functions of these gates can be achieved by using equations 17 through 22 in Supplemental Materials A.2. Except for 19 and 22, all equations can be modified into secure ones based on the proposed secure protocols. As for Eq.19 and 22, it seems obvious that the key issue is to realize the secure computation of the function $tanh(x)$. Fortunately, the $tanh(x)$ can be represented as the function of the sigmoid function $\sigma(x)$ through the following derivation.

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{2}{1 + e^{-2x}} - 1 = 2\sigma(2x) - 1. \quad (4)$$

By using the **SecSig** protocol, the secure version of $tanh(x)$ can be achieved, and denoted as **SecTan**(x).

A. Secure Forward Propagation

The computation tasks in the input gate contain all operations that appeared in both the forget gate and output gate. For the sake of simplicity, we only take the input gate as an example to illustrate how to employ the additive secret sharing technique to calculate the related tasks securely. First, $\mathcal{P}_1$ and $\mathcal{P}_2$ cooperatively to decide which is input data. Concretely, given time t, $\mathcal{P}_1$ and $\mathcal{P}_2$ separately calculate

$$(\widetilde{i}_t, \widehat{i}_t) = \textbf{SecSig}\left(W_i * [h_{t-1}, x_t, C_{t-1}] + b_i\right),$$

In the next step, a new candidate C'_t is created to decide which information in the input data will be added to the current state. To securely achieve it, $\mathcal{P}_1$ and $\mathcal{P}_2$ need to generate

$$(\widetilde{C}'_t, \widehat{C}'_t) = \textbf{SecTan}\left(W_C * [h_{t-1}, x_t] + b_c\right).$$

Based on the above steps, $\mathcal{P}_1$ and $\mathcal{P}_2$ jointly create a secure update for the current state C_t, namely,

$$(\widetilde{C}_t, \widehat{C}_t) = \textbf{SecAdd}(f_t \circ C_{t-1}, i_t \circ C'_t).$$

The Hadamard product "$\circ$" is typically performed by the pairwise multiplication of two input matrices. Therefore, $\mathcal{P}_i$ can correctly carry out the "$\circ$" operation over the ciphertext data by calling **SecMul** protocol. Hereafter, we denote the secure protocol of the "$\circ$" operation as **SecHad**.

Similarly, the edge servers also finish all computation tasks related to the forget/output gate, without knowing about the plaintext data.

B. Secure Backward Propagation

In CLSTM, the total loss E at time t can be represented as the function of the hidden cell h_t. Therefore the error δ_{h_t} for h_t is available, where δ_{h_t} is equal to the partial derivative of E, namely, $\frac{\partial E}{\partial h_t}$. By applying the chain rule, we can obtain

$$\delta_{o_t} = \frac{\partial E}{\partial o_t} = \frac{\partial E}{\partial h_t} \cdot \frac{\partial h_t}{\partial o_t} = \delta_{h_t} \cdot \frac{\partial h_t}{\partial o_t},$$

Based on Eq.22 in Supplemental Materials A.2, we rewrite the above equation as:

$$\delta_{o_t} = \delta_{h_t} \circ tanh(C_t), \quad (5)$$

Following the same process for f_t, i_t, and C'_t, we can get

$$\delta_{f_t} = \delta_{h_t} \circ (1 - tanh^2(C_t)) \circ o_t \circ C_{t-1}, \quad (6)$$

$$\delta_{i_t} = \delta_{h_t} \circ (1 - tanh^2(C_t)) \circ o_t \circ C'_t, \quad (7)$$

$$\delta_{C'_t} = \delta_{h_t} \circ (1 - tanh^2(C_t)) \circ o_t \circ i_t, \quad (8)$$

$$\delta_{h_{t-1}} = \delta_{f_t} * W_{hf} + \delta_{i_t} * W_{hi} + \delta_{C'_t} * W_{hc} + \delta_{o_t} * W_{ho}, \quad (9)$$

where o_t, f_t, i_t, C'_t can be obtained from the forward propagation process.

Using the **SecHad**, **SecAdd**, **SecMul**, **SecCon** protocols, we can allow ESs to calculate these equations 5 through 9 on the obfuscated shares. As a result, ESs can calculate the partial derivative of the total loss with respect to all weights and biases over ciphertext data as follows.

$$(\nabla\widetilde{W}_{xC'}, \nabla\widehat{W}_{xC'}) = \sum_{t=1}^{T} \textbf{SecCon}(\textbf{SecHad}(\delta_{C'_t}, \varphi(\overline{C}'_t)), x_t),$$

$$(\nabla\widetilde{W}_{hC'}, \nabla\widehat{W}_{hC'}) = \sum_{t=1}^{T} \textbf{SecCon}(\textbf{SecHad}(\delta_{C'_t}, \varphi(\overline{C}'_t)), h_{t-1}),$$

$$(\nabla\widetilde{b}_c, \nabla\widehat{b}_c) = \sum_{t=1}^{T} \textbf{SecHad}(\delta_{C'_t}, \varphi(\overline{C}'_t)),$$

where T denotes the time step. $\varphi(x)$ represents the derivative of the function $tanh(x)$, viz., $\varphi(x) = tanh'(x) = (1 - tanh^2(x))$. The overbar represents the input of tanh in the corresponding equation. Similarly, $\mathcal{P}_i$ can securely calculate $\nabla W_{h\mu}, \nabla W_{C\mu}, \nabla W_{x\mu}, \nabla b_\mu$, where μ belongs to the set of $\{f, i, o\}$. Following, the weights and biases can be securely updated by using the same update equations as ACE part.

5.3 Secure Anomaly Evaluation

In the plaintext domain, when the model is trained, the regularity score s used to evaluate abnormality of video frame x is calculated as follows.

$$s(x) = 1 - \frac{e(x) - min_x e(x)}{max_x e(x)}, \quad (10)$$

$$e(x) = \|x - f_W(x)\|_2, \quad (11)$$

where $f_W(\cdot)$ denotes the reconstruction operation using the trained model weights W. $e(x)$ represents the reconstruction error between the original frame x and the reconstructed frame $f_W(x)$. $\|\cdot\|_2$ indicates the Euclidean distance.

Based on the above equations, the larger $e(x)$ is, the lower the regularity score $s(x)$ is, which means that video frame x has a higher probability of being anomalous. On the contrary, the frame with higher regularity score is always determined to be normal. As Eq. 10 and 11 only involve the simple elementary operations, they are easily modified as the following secure versions using the proposed additive secret sharing based interactive protocols.

$$\begin{aligned}(\widetilde{s}(x), \widehat{s}(x)) = \textbf{SecAdd}(1, -\textbf{SecMul}(\textbf{SecAdd}(e(x), \\ -\textbf{SecMin}(e(x))), \textbf{SecRec}(\textbf{SecMax}(e(x))))),\end{aligned}$$

$$\begin{aligned}(\widetilde{e}(x), \widehat{e}(x)) = \textbf{SecSqr}(\textbf{SecMul}(\textbf{SecAdd}(x, -f_W(x)), \\ \textbf{SecAdd}(x, -f_W(x))))\end{aligned}$$

where **SecMin** denotes minimum secure protocol, which can be obtained just by choosing the reverse result in **SecMax** protocol. The **SecSqr** is a secure protocol of being

used to calculate the square root of a positive number. Similar to the **SecRec** protocol, the **SecSqr** protocol is achieved by the Newton-Raphson iterative method, where $x_{k+1} = \frac{1}{2}(x_k + \frac{u}{x_k})$ is instead of Eq.23 (in Supplemental Materials B.4) in **SecRec** protocol.

5.4 Functional extension for our SecureAD

We list three functional extensions for SecureAD as follows.

Extension for multiple COs: In the previous section, SecureAD is mainly designed for the setting: a CO with multiple AUs. In fact, our SecureAD is also suitable for multiple COs, which attributes to the key-free characteristic of the additive secret sharing technique that is used for the construction of all non-interactive/interactive protocols in SecureAD. In addition, the proposed access control mechanism supports multiple COs scenarios.

Extension for multiple ESs: As described in Section 4, all secure protocols are based on the protocols **SecAdd** and **SecMul**. It indicates that these two protocols determine the scalability of our SecureAD to $n(n \geq 3)$ ESs. Since **SecAdd** is performed locally, it can be easily extended to the n ESs setting. As for **SecMul**, we can directly employ the multiparty technique in [12] to improve it so that SecureAD meets the needs of multiple ESs.

Extension for other applications: As the proposed protocols can achieve secure CNN model training, SecureAD with appropriate modifications can be applied to other CNN-based secure computer tasks, such as image classification, instance segmentation, object localization, etc.

6 Analysis of our SecureAD

In this section, we first illustrate the correctness analysis of SecureAD, and then give its security and performance.

6.1 Correctness Analysis

In SecureAD, all involved linear and nonlinear are calculated by using the additive secret sharing technique. To be special, we design some secure protocols with the additive property, and then elaborately combine them to achieve all computation tasks in SecureAD. Therefore, the correctness of SecureAD completely depends on whether these protocols introduce errors or not.

Theorem 1. *The difference of the video anomaly detection results between SecureAD and its plaintext version is negligible.*

Proof: Considering whether the error exists, the security protocols in SecureAD can be divided into two types. The *first type* has the error-free characteristic. This type of protocol includes the **SecAdd**, **SecMul**, **SecXor**, **SecCom**, **SecMax**, **SecCon**, and **SecHad** protocols. Among them, the former two protocols are correct, which are concluded in [13]. As described in Section 5, the latter five protocols are a linear combination of the **SecAdd** and **SecMul** protocols in an error-free manner. Therefore, the last five protocols are also correct. The *second type* consists of four sub-protocols, **SecExp**, **SecRec**, **SecSig**, and **SecTan**. Different from the first type, this type of protocol will introduce approximation errors, which mainly are caused by the following two factors. *One* is to use n-th order Maclaurin expansion to approximate the natural exponential function e^u. According to Taylor's expansion theorem, the error produced by this approximation method can be denoted as the remainder in Lagrange form as follows.

$$R_n(u) = e^u - \sum_{k=0}^{n} \frac{1}{n!} \cdot u^n = \frac{e^\xi}{(n+1)!} u^{n+1}, \tag{12}$$

where $\xi \in (0, u)$. The above equation indicates that the $R_n(u) \to 0$ when $n \to \infty$ and $u \in (0, 1)$. It means that the approximation error can be controlled at an arbitrary low level as long as n is big enough. In fact, we only need set n to 13 such that the high order of accuracy for e^u with a negligible error not less than 10^{-10} can be achieved.

Another is to employ the Newton-Raphson iterative method to approximate $1/u$. Let the $(n+1)$-th and n-th iterative errors be $\varepsilon_{n+1} = 1/u - x_{n+1}$ and $\varepsilon_n = 1/u - x_n$, respectively. In theory, these two errors satisfy the following.

$$|\varepsilon_{n+1}| = \frac{|f''(\xi_n)|}{2|f'(x_n)|} \cdot \varepsilon_n^2, \tag{13}$$

where ξ_n is in between x_n and $1/u$. The symbol "||" takes absolute value and $f(x) = 1/x - u$. Eq. 13 implies that the Newton-Raphson method can achieve quadratic convergence. If only the initial estimate x_0 is in between 0 and $2/u$, $\lim_{n\to\infty} x_n = 1/u$ holds.

As described in Section 5, our SecureAD is constructed based on the various linear combination of the above two types of protocol. Consequently, the accuracy of our SecureAD can infinitely close to that of its corresponding plaintext framework only if we guarantee the errors of the second type of protocol are at a negligible level. □

Theorem 2. *The proposed user access authentication is correct under the honest-but-curious model, provided that a user is authorized.*

Proof: In SecureAD, we employ the BF technique to build an access authentication mechanism. Due to the BF having no false negatives, each attribute of an authorized user can be ensured to correctly match with the access policies made by the corresponding COs. Besides, ESs follow the pre-set protocols to carry out their computation tasks under the *honest-but-curious* model. It means that they cannot tamper the intermediate or final results such that the access authentication can be correctly calculated. Therefore, our SecureAD can assure that the access authentication is successfully performed when a user is authorized. □

Remark: Although the **Theorem 2** can guarantee that authorized users are successfully authenticated, there are false positives during the BF usage. It indicates that an unauthorized user may be verified through. This issue can be solved by performing the following steps. In the first step, we can select proper parameters to reduce the false positive probability largely. It is shown in [19] that the false positive probability can be minimized as $2^{-k} \approx 0.618^{m/n}$ on the condition of $k = ln2 \cdot (m/n)$, where all meanings of the tunable parameters k, m, n refer to Subsection 2.3. In the second step, the false positive probability can be further reduced by allotting multiple BFs to each attribute. Let ω

be the number of BF for a given attribute. The false positive probability of the current attribute will fall to

$$Pr(FP) = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^{k\omega}. \quad (14)$$

Clearly, the false positives are statistically insignificant when ω is large enough, which is verified in the latter experiments.

6.2 Security Analysis

In this subsection, we prove the security of sub-protocols and then give the security analysis of our SecureAD. To better analyze the security of sub-protocols, we first formally present the secure definition and related elementary lemmas of a share-based computation protocol [13], [14], [15].

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

Lemma 1. *A protocol is perfectly simulatable if all its sub-protocols are perfectly simulatable* [20].

Lemma 2. *If a random element r is uniformly distributed on $\mathbb{Z}_n$ and independent from any variable $x \in \mathbb{Z}_n$, then $r \pm x$ is also uniformly random and independent from x* [21].

Theorem 3. *The **SecCom** protocol in Subsection 4.2 is secure to compare which one on two private numbers is larger under the honest-but-curious model.*

Proof: In the **SecCom** protocol, the ES $\mathcal{P}_1$ has the $View_1 = \{u_1, v_1, a_1, r_1^{(k)}, s_1^{(k)}, t_1^{(k)}, \alpha_1^{(k)}, \beta_1^{(k)}\}$, where $k \in [0, \varrho - 1]$. Since all elements of the set $\{u_1, v_1, a_1, r_1^{(k)}\}$ is uniformly random, $s_1^{(k)}, t_1^{(k)}, \alpha_1^{(k)}, \beta_1^{(k)}$ follow the uniform random distribution according to the **Lemma 2**, where $t_1^{(0)} = 0$ is an exception. Besides, it can be found that $\mathcal{P}_1$'s $Output_1$ $d_1^{(\varrho-1)} = s_1^{(\varrho-1)} \oplus r_1^{(\varrho-1)} \oplus t_1^{(\varrho-1)}$ is also uniformly random, where $t_1^{(\varrho-1)} = \beta_1^{(\varrho-2)} \oplus \alpha_1^{(\varrho-2)}$. As a result, $View_1$ and $Output_1$ are computationally indistinguishable for the adversary $\mathcal{A}$. The same case for $\mathcal{P}_2$. □

Theorem 4. *The **SecMax** protocol proposed in Subsection 4.3 is secure to find the maximum of a series of numbers under the honest-but-curious model.*

Proof: As described in Subsection 4.3, the **SecMax** protocol takes the **SecCom** protocol as sub-protocol. And also, all input numbers are uniformly random at each edge server. Therefore, it can be concluded that the **SecMax** protocol is secure under the honest-but-curious model based on **Theorem 3** and **Lemma 1**. □

Theorem 5. *The protocols **SecRec**, **SecExp**, **SecSig**, and **SecTan** are secure under the honest-but-curious model.*

Proof: Please refer Supplemental Materials C for a security proof. □

Theorem 6. *The protocols **SecXor**, **SecCon**, and **SecHad** are secure under the honest-but-curious model.*

Proof: Please refer Supplemental Materials C for a security proof. □

Theorem 7. *Our SecureAD framework is secure under the honest-but-curious model.*

Proof: Essentially, our SecureAD is composed of all sub-protocols as previously mentioned, which are described detailedly in Subsection 5.2. It indicates that SecureAD can be viewed as a big protocol, which is a combination of the previous sub-protocols. In terms of **Lemma** 1, we can guarantee the security of our SecureAD because that all related sub-protocols have been proved to be secure under the *honest-but-curious* model. □

Theorem 8. *Our SecureAD framework is secure against the adversaries $\mathcal{A}_{i,j}^*(i \in \{1,2,3\}, j \in \{1,2\})$ defined in Subsection 3.3.*

Proof: In our security model, the adversary $\mathcal{A}_{i,j}^*$ is assumed to have an ability to eavesdrop the communication link between the party i and the edge server j. It means that $\mathcal{A}_{i,j}^*$ can get all transmitted data from the link. Besides, all data stored in the ES j may also be available to $\mathcal{A}_{i,j}^*$ when the ES j is compromised by $\mathcal{A}_{i,j}^*$. In SecureAD, these data are meaningless shares split by using the additive secret sharing technique. Since $\mathcal{A}_{i,j}^*$ does not obtain all two shares from the same data, $\mathcal{A}_{i,j}^*$ cannot reconstruct the corresponding plaintext data. Due to compromising with the ES j, $\mathcal{A}_{i,j}^*$ may also obtain the intermediate ciphertext data transmitted by another ES during CNN model training and anomaly detection process. Based on the knowledge in Section 5, it is obvious that these intermediate data are generated mainly by executing the **SecCom** or **SecMax** or **SecRec** or **SecExp**. According to **Theorem** 3 through **Theorem** 5, these intermediate data are uniformly random, which cannot be decrypted by $\mathcal{A}_{i,j}^*$. Besides, additive secret sharing technique allows different user attribute shares/video frame shares to correspond to the same user/frame. So, the non-deterministic characteristic can guarantee request unlinkability, which disables statistical attacks.

Based on the above analysis, our SecureAD is secure against the adversaries defined in the security model. □

6.3 Performance Analysis

We evaluate SecureAD on benchmarking dataset: Avenue [22], which is one of the most commonly used databases for video anomaly detection. The dataset comprises of 37 video clips, where 16 video clips are for the training and others are used for the testing. The evaluations are carried out on three Lenovo laptops, each of which is equipped with an I5-7200 2.5GHz processor and 8GB Memory running windows 7-64bit. Specifically, a laptop is used to simulate CO or RP, and another two laptops for edge servers. In the following experiments, the number of iteration and the remainder are set to 30 and 10^{-10}, respectively. These two parameters are discussed in Supplemental Materials D.

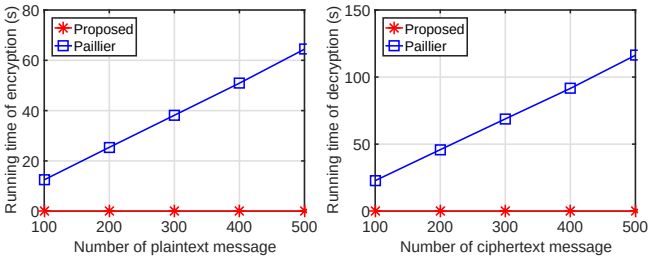
6.3.1 Performance evaluation of encryption

It is well known that Paillier cryptosystem[23] is a public-key cryptsystem due to its probabilistic encryption and additively homomorphic property. At present, it is widely used to achieve the privacy-preserving machine learning. Therefore, we made a performance comparison between our scheme with Paillier cryptosystem, in which the public key

(PK) is set to $N = 2048$-bit in consideration of the security strength of Paillier crptosystem. The implementation results listed in TABLE 3 show that the number of Paillier ciphertext (Ctxt) bits is 4096, but only 2049 bits in our scheme for a same 2048-bit message m. The main reason is that Paillier Ctxt is generated under modulus N^2, and our Ctxt is composed of m's two random secret shares, each of which has the same bit length with m. It is not surprising to find from TABLE 3 that PK is 0 in our scheme, because the proposed encryption is keyless. The same reason is for the key generation (KeyGen). As shown in TABLE 3, the computation costs of the addition (Add) and scalar multiplication (SMul) operations in our scheme are substantially below those of Paillier encryption, reducing two orders of magnitude.

TABLE 3
Implementation results.

Schemes	Ctxt	PK	KeyGen	Add	SMul
Paillier	4096-bit	2048-bit	$670.80ms$	$62.40\mu s$	$42.12\mu s$
Ours	2049-bit	0	0	$0.20\mu s$	$0.23\mu s$



(a) (b)

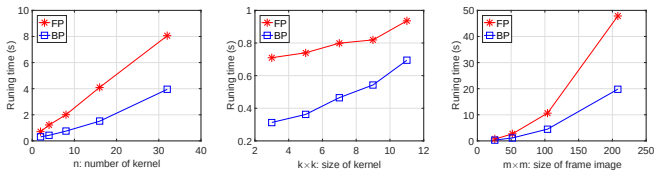
Fig. 3. Performance analysis in different algorithms: (a) Computation costs in the encryption process; (b) Computation costs in the decryption process.

To further demonstrate the superiority, the comparison of the computation costs for the encryption and decryption is also conducted. As we can see from Fig. 3, the running time of both encryption and decryption in Paillier cryptosystem rapidly goes up with the increase of the number of the corresponding messages. This is mainly because that it has higher computation costs and ciphertext expansion rate than that of our scheme. Moreover, Fig. 3 illustrates that when adding the number of messages, the computation costs of these two processes in our scheme are almost the same and are close to 0. Specifically, when the number of the plaintext/ciphertext messages is 10^7, the entire costs of the encryption/decryption process $218.4/31.2$ milliseconds only. It means that our scheme is more efficient.

Furthermore, there are strong correlations between adjacent pixels in different directions, which contain horizontal, vertical, diagonal directions. In general, the effective reduction of these correlations is a critical indicator for a good encryption algorithm. Therefore, we introduce the correlation coefficient to evaluate these three correlations in our scheme. It can be defined as $cov(x,y)/(\sqrt{D(x)}\sqrt{D(y)})$, where $cov(\cdot,\cdot)$ and $D(\cdot)$ denote covariance and variance functions, respectively. If r is close to 0, the weak correlation will be obtained. If r is close to 1, the strong correlation appears. The comparison with the plaintext ones is listed in TABLE 4, where the five classic gray images (Man, Lena, Boat, Bridge, Baboon) sized of 512×512 are used to evaluate. Note that an image by our scheme is encrypted into the two shares with the same size as that of the original image. It is easy to find that our scheme can effectively eliminate these three correlations.

6.3.2 *Performance of our SecureAD*

We first analyze the effect of the convolutional operation on the efficiency of CNN, which is typically performed on convolutional operation. Take CAE* (described in Supplemental Materials D) as an example, we illustrate that three factors related to the convolutional operation: the number of kernel, the size of kernel, and the size of frame image, are how to affect the CNN's training time.



(a) (b) (c)

Fig. 4. Performance analysis on convolutional operation for the forward propagation (FP) and backward propagation (BP). (a) Effect of n on training time ($m = 26, k = 3$); (b) Effect of k on training time ($m = 26, n = 2$); (c) Effect of m on training time ($k = 3, n = 2$).

It is clear from Fig. 4 that either FP or BP takes more computational time as n or k or m increases. Given a video, the size of frame image is always fixed as a constant. In this case, n and k will affect the computational costs of the convolutional operation, which indirectly has an impact on the efficiency of secure CNN training or testing. Compared with the size of kernel, n has a bigger impact on the CNN's performance. This is because that n is generally far larger than k in the realistic scenes.

In SecureAD, we adopt the construction of CNN in [6] to set up the privacy-preserving video anomaly detection system. TABLE 5 presents the architecture of our CNN and gives some necessary configuration information: the number of the kernel, kernel size, stride. The last column shows the number of the weight parameters in each layer, which can be calculated from the configuration information of the corresponding layer. As stated in Section 2, our CNN includes the two parts: CAE and CLSTM. The former is used to capture the spatial information of anomalous events in video streams. And it consists of an encoder and a decoder, which are separately represented by the first two layers and the last two layers in TABLE 5. The latter is capable of tracing the temporal information of anomalous behaviors. There are three different configuration parameters for CLSTM in SecureAD. For the sake of distinction and follow-up discussion, we denote them as CLSTM-I, CLSTM-II, and CLSTM-III, which correspond to the third, fourth, and fifth layers in TABLE 5, respectively. It was well known that a CNN model can be usually achieved through several rounds of FP and BP, which is commonly called the model training. After the CNN model training, only forward propagation is

TABLE 4
The correlation coefficients of original image and encrypted versions.

Image	Original image			Share for $\mathcal{P}_1$			Share for $\mathcal{P}_2$		
	Horizontal	Vertical	Diagonal	Horizontal	Vertical	Diagonal	Horizontal	Vertical	Diagonal
Man	0.9673	0.9600	0.9384	−0.0144	−0.0011	0.0273	−0.0013	−0.0018	−0.0165
Lena	0.9847	0.9769	0.9617	−0.0183	−0.0223	0.0039	−0.0415	−0.0144	0.0120
Boat	0.9715	0.9392	0.9145	0.0178	−0.0510	−0.0029	−0.0196	−0.0066	−0.0185
Bridge	0.9252	0.9422	0.9075	−0.0114	0.0104	−0.0027	−0.0389	−0.0099	−0.0034
Baboon	0.7764	0.8672	0.7531	0.0056	−0.0017	−0.0037	−0.0123	−0.0163	0.0043

used to test the performance of anomaly detection. Therefore, we focus on the performance analysis of FP and BP in the following experiments.

TABLE 5
Our convolutional neural network architecture.

Layer (type)	# of kernel	Kernel size	Stride	# of parameter
Convolution	128	11×11	4	15616
Convolution	64	5×5	2	204864
CLSTM	64	3×3	1	295168
CLSTM	32	3×3	1	110720
CLSTM	64	3×3	1	221440
Deconvolution	128	5×5	2	204928
Deconvolution	1	11×11	4	15489

Stride: The step size of the kernel movement; "#": The number.

Efficiency evaluation: In TABLE 6, we first compare the performance of two stages (encoding and decoding) composed of CAE, where the batch size is 10, and the bit-length ℓ of data is 32. It is evident that the computational complexity of BP is higher than that of FP, no matter which stage. We attribute it to the fact that more convolutional operations are carried out in BP. Since convolution is a local and dominant operation in CNN, the number of convolution operations could account for the computational costs in CNN. Also, we observe that more time is consumed in the decoding stage than the encoding stage during FP or BP process. This is because massive padding operations significantly enlarge the size of input data, resulting in the increasing amount of convolution operations, and then raising computational overheads.

TABLE 6
Comparison of the runtime and message size on different stages in CAE part of our CNN.

Stages	Forward propagation		Backward propagation	
	Runtime (s)	RP→ ES (MB)	Runtime (s)	RP→ ES (MB)
Encoding	184.464	4.995	967.846	0.678
Decoding	649.635	0.339	1241.047	5.334

In **SecMul** protocol, the additive shares of three random numbers need to be transmitted from RP to each ES, in which the network bandwidth is required to transit. As a convolutional operation involves several multiplications, its secure version **SecCon** protocol is performed by using more bandwidths. Assume that the size of a kernel is $k \times k$. In theory, RP needs to send messages with a total length of $3k^2\ell$ bits to each ES just for secure convolution operation at a time. Based on this conclusion, around 4289.632 MB bandwidths are consumed in the first layer of our CNN. Obviously, the whole CNN incurs more communication overheads, generally with GB-scale.

Let the size of the feature maps in the $(l-1)^{th}$ layer be $h \times h$. To reduce the communication costs, we only provide three random maps of size $h \times h$ to assist ES to securely carry out the multiplication operations in the l^{th} layer, where the relationship of the three numbers at the same position of three random maps equals that of a, b, c in **SecMul**. Thus, the strategy can offer three random numbers for each position in feature maps. Note that, given a layer in CAE, three random numbers are shared by different feature maps, and are regenerated to each batch sample for better safety. As listed in TABLE 6, the strategy of random maps greatly decreases communication overloads. For the same first layer, RP equipped with random maps only sends around 4.72 MB to each ES, which is far lower than the original 4289.632 MB. Meanwhile, we observe that the communication costs of the encoding stage are much more than the decoding stage in FP. It can be explained that the size of the feature map at the encoding stage is relatively bigger than that at the decoding stage. The same reason can be used to explain the case existed in BP.

TABLE 7
Comparison of the runtime and message size on different gates in CLSTM-I with 64 kernels and 64 channels.

Gates	Forward propagation		Backward propagation	
	Runtime (s)	RP→ ES (MB)	Runtime (s)	RP→ ES (MB)
Forget Gate	88.707	0.186	71.734	0.371
Input Gate	148.231	0.309	129.301	0.681
Output Gate	89.988	0.433	72.862	0.495

Furthermore, we evaluate the efficiency of CLSTM part in SecureAD. TABLE 7 illustrates a comparison of the runtime and message size on different gates in CLSTM-I, in which the time step is equal to 10. Compared to the above CAE part, FP in CLSTM-I occupies more computational time than BP. It can be deduced from the computational equations involved in the three gates that the scale of the convolutional operations in FP is larger than that in BP. The other reason is that the same part $\delta_{h_t} \circ (1 - tanh^2(C_t)) \circ o_t$ in equations 6 through 8 only needs to be calculated once in BP. TABLE 7 also shows the compared results of the communication costs over different gates and stages. Since the

size of the feature maps in CLSTM-I is relatively small, we here assign a set of three random maps for each convolution or Hadamard product appeared in the relevant equations to ensure that the intermediate results in each ES are random enough. In terms of computation costs and communication overloads, we can observe from TABLE 8 and 9 that the correlative relationship between the different gates or stages in CLSTM-II/CLSTM-III is consistent with that in CLSTM-I.

TABLE 8
Comparison of the runtime and message size on different gates in CLSTM-II with 32 kernels and 64 channels.

Gates	Forward propagation		Backward propagation	
	Runtime (s)	RP→ ES (MB)	Runtime (s)	RP→ ES (MB)
Forget Gate	26.807	0.186	17.731	0.371
Input Gate	45.116	0.309	33.063	0.681
Output Gate	27.654	0.433	18.206	0.495

TABLE 9
Comparison of the runtime and message size on different gates in CLSTM-III with 64 kernels and 32 channels.

Gates	Forward propagation		Backward propagation	
	Runtime (s)	RP→ ES (MB)	Runtime (s)	RP→ ES (MB)
Forget Gate	53.872	0.186	49.584	0.371
Input Gate	88.554	0.309	86.969	0.681
Output Gate	55.106	0.433	49.604	0.495

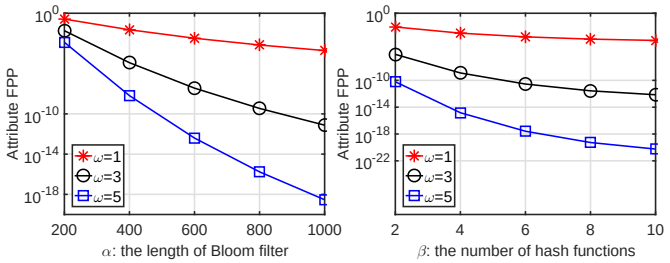
TABLE 10
Ratio of different operations in our CNN.

Operation type	CAE part	CLSMT-I	CLSMT-II	CLSMT-III
Convolution	98.39%	97.92%	97.41%	97.43%
Activator	1.55%	1.71%	2.11%	2.13%
Others	0.06%	0.37%	0.48%	0.44%

The results presented in the above four tables indicate that a major concern for our SecureAD is the high cost of time. To address this issue. We made statistics on computational costs of some operations in SecureAD. TABLE 10 shows that secure convolutional operation in any part of our whole CNN has the largest proportion for time consumption, and then activator operations (contains **SecSig** and **SecTan**), and other operations (contains **SecAdd**, **SecHad** and so on) least. Fortunately, the convolutional operations in the same layer are completely independent of each other, which means that we can use the existing mature parallel techniques to reduce the time consumption drastically.

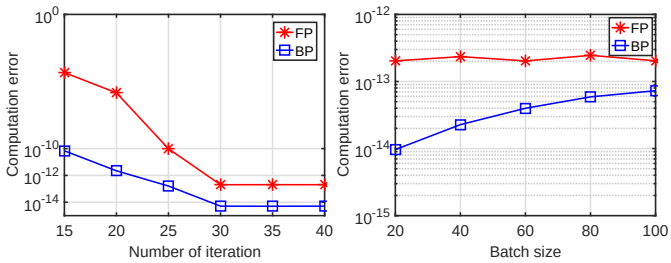
Fig. 5(a) and 5(b) illustrate the effect of the length of Bloom filter α and the number of hash functions β with different ω on attribute false positive probability (FPP). We observe that the lower attribute FPP will be achieved when using the larger α or β under the same conditions. Additionally, as the number of Bloom filter ω increases, the lower FPP can be obtained, theoretically, would tend to zero, which is verified by Eq. 14. For example, when $\alpha = 1000$, $\beta = 10$, and the number of users' attribute value $\gamma = 50$, FPP with $\omega = 5$ is only 5.7×10^{-21}, which is far lower than 8.9×10^{-5} with $\omega = 1$. However, it will increase the communication costs between RP and ES. There needs to be a trade-off between communication costs and FPP. In fact, γ is also a factor to affect FPP. Different from factors α, β, it is of proportional relation with FPP.

In SecureAD, user access control only involves addition and multiplication operations, so the runtime to authenticate a user is relatively slow. Specifically, when $\alpha = 1000$, $\beta = 10$, $\gamma = 50$ and $\omega = 5$, each edge server will only consume on average around 46 ms to perform its computation task for a user with 50 attributes. In real scenarios, however, the number of user attribute is no more than 20 [24].



(a) (b)

Fig. 5. Effect of α and β on attribute FPP. ω denotes the number of Bloom filters. (a) The attribute FPP as a function of α ($\beta = 7$, and $\gamma = 50$); (b) The attribute FPP as a function of β ($\alpha = 1000$, and $\gamma = 50$).



(a) (b)

Fig. 6. Effect of our SecureAD on the accuracy of video anomaly detection. (a) The computation error as a function of iteration numbers; (b) The computation error as a function of batch size.

Accuracy evaluation: In SecureAD, the approximate processes of the **SecExp** and **SecRec** protocols may affect the accuracy of the detection system. Considering this, we also evaluate the computation error introduced by SecureAD at FP and BP stages. As shown in Fig. 6(a), we find that the computation error can still be controlled at a negligibly low level, when giving an appropriate iteration number, such as 30. Additionally, in order to reduce computation error, we introduce Gaussian distribution into the weight parameter initialization to accelerate the rate of our CNN's convergence, which avoids the rapid accumulation of computation error during CNN training. Meanwhile, we also analyze the effect of the batch size on the computation error. In Fig. 6(b),

it can be observed that changing the batch size will hardly influence the computation error at FP stage, where the number of iteration is 30. Relatively speaking, the computation error at BP stage shows an upward tendency when the batch size increases. However, the increase in the computation error is relatively trivial. When the batch size is 100, the corresponding computation error is only 7.28×10^{-14}, which is almost unable to affect the accuracy of anomaly detection. It shows that SecureAD is valid and may achieve almost the same accuracy as its plaintext counterpart with a negligible computation error.

7 Related Work

In the plaintext domain, anomaly event detection is viewed as one of the most challenging issues due to the diversity and complexity of anomaly types [22], [25], [26], [27], [28]. At present, the research on the video anomaly has attracted extensive concerns, and many related techniques are developed successively [29], [30], [31]. As reviewed in [32], [33], the early works for the video event detection are mainly classified into two types, trajectory-based methods [25], [34] and non-tracking methods [4], [5]. Following the success of the deep learning techniques in various computer vision tasks [35], the CNN-based anomaly detection has also raised considerable attention recently [29], [36].

In the encrypted domain, privacy-preserving machine learning technique dates back to [37], which trains the classifier over encrypted individual data records. After that, various types of secure machine learning approaches are proposed. The encryption methods adopted by these approaches can be mostly divided into three types, namely, differential privacy (DP), multi-party computation (MPC), and HE. DP applies noise to conceal partially individual data such that the statistic information on the dataset can be calculated [38]. With DP, the final statistic accuracy is always lower than that of the original plaintext data due to the introduction of the noise. MPC is a cryptographic primitive of securely jointing multiple parties to calculate the same function, in which the input data of any party is not leaked from each other. Using garbled circuits (GC) [39] to investigate MPC can achieve any function calculation [40], [41], but high computation complexity incurs low efficiency and poor reusability. Therefore, most of GC-based MPC is still left in theoretical study, which is indicated in [42]. Recently, the additive secret sharing based secure MPC has been proposed to process some tasks related to a neural network, such as feature extraction [15], and speech recognition [14]. However, the authors of [15] cannot provide a solution to protect the privacy of the weight parameters. And the scheme in [14] is limited to the full connected network. HE is an asymmetric encryption algorithm, which is generally used to encrypt data while allowing users to perform computations on encrypted data before decrypting. At present, it has been widely studied in neural networks for maintaining the privacy of input data, model, returned results [8], [9], [43], [44], [45], [46]. However, high computation costs and encrypted data expansion limit HE to be practical.

To the best of our knowledge, the privacy-preserving CNN-based framework for video anomaly detection has not been proposed yet. Most relevant works mainly focus on motion detection and tracking over encrypted surveillance videos [16], [47], [48], [49]. The scheme in [50] can be used to detect anomaly event, but its encryption algorithm is not suitable for CNN. Moreover, the concept of anomaly detection is also used to find network traffic [51], [52] and malicious behavior [53]. In this paper, we develop a novel framework SecureAD to offer a solution for detecting event anomalies over encrypted videos. In SecureAD, the content owners only outsource data shares to two ESs. When obtaining these data, ESs can train the CNN model and offer detection results, without compromising the privacy and security concerns. And also, SecureAD can support attribute based access control in multi-user scenarios.

8 Conclusions

We presented a framework called SecureAD for privacy-preserving video anomaly detection by combining the convolutional neural networks and secret sharing technique. A series of secure novel calculation protocols were designed to allow the edge servers to perform the CNN models training and anomalous evaluations. In addition, we also developed a new attribute-based access control mechanism by using the Bloom filter technique. This mechanism could enable servers to identify the legality of a query user, without comprising the privacy of users' attributes. Furthermore, the theoretical analysis and practical simulations showed that SecureAD could be effectively performed with a negligible computation error.

Acknowledgment

The authors thank the Associate Editor and reviewers for their constructive and generous feedback. This work was supported by the National Natural Science Foundation of China (No. U1804263, No. 61702105), the National Research Foundation, Prime Ministers Office, Singapore under its Strategic Capability Research Centres Funding Initiative, and Singapore Ministry of Education under Research Grant MOE2016-T2-2-014(S).

References

[1] L. Tian, H. Wang, Y. Zhou, and C. Peng, "Video big data in smart city: Background construction and optimization for surveillance video processing," *Future Generation Computer Systems*, 2018.

[2] Y. Zhao, B. Deng, C. Shen, Y. Liu, H. Lu, and X.-S. Hua, "Spatio-temporal autoencoder for video anomaly detection," in *Proceedings of the 25th ACM international conference on Multimedia*. ACM, 2017, pp. 1933–1941.

[3] K. Pawar and V. Attar, "Deep learning approaches for video-based anomalous activity detection," *World Wide Web*, pp. 1–31, 2018.

[4] B. Zhao, L. Fei-Fei, and E. P. Xing, "Online detection of unusual events in videos via dynamic sparse coding," in *CVPR 2011*. IEEE, 2011, pp. 3313–3320.

[5] Y. Cong, J. Yuan, and J. Liu, "Sparse reconstruction cost for abnormal event detection," in *CVPR 2011*. IEEE, 2011, pp. 3449–3456.

[6] Y. S. Chong and Y. H. Tay, "Abnormal event detection in videos using spatiotemporal autoencoder," in *International Symposium on Neural Networks*. Springer, 2017, pp. 189–196.

[7] Z. Yan, J. Xue, and C. W. Chen, "Prius: Hybrid edge cloud and client adaptation for http adaptive streaming in cellular networks," *IEEE transactions on circuits and systems for video technology*, vol. 27, no. 1, pp. 209–222, 2017.

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

[9] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in *International Conference on Machine Learning*, 2016, pp. 201–210.

[10] Y. Aono, T. Hayashi, L. Wang, S. Moriai *et al.*, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, pp. 1333–1345, 2017.

[11] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Advances in neural information processing systems*, 2015, pp. 802–810.

[12] D. Beaver, "Efficient multiparty protocols using circuit randomization," in *Annual International Cryptology Conference*. Springer, 1991, pp. 420–432.

[13] Z. Ma, Y. Liu, X. Liu, J. Ma, and K. Ren, "Lightweight privacy-preserving ensemble classification for face recognition," *IEEE Internet of Things Journal*, 2019.

[14] Z. Ma, Y. Liu, X. Liu, J. Ma, and F. Li, "Privacy-preserving outsourced speech recognition for smart iot devices," *IEEE Internet of Things Journal*, 2019.

[15] K. Huang, X. Liu, S. Fu, D. Guo, and M. Xu, "A lightweight privacy-preserving cnn feature extraction framework for mobile sensing," *IEEE Transactions on Dependable and Secure Computing*, 2019.

[16] H. Cheng, H. Wang, X. Liu, Y. Fang, M. Wang, and X. Zhang, "Person re-identification over encrypted outsourced surveillance videos," *IEEE Transactions on Dependable and Secure Computing*, 2019.

[17] X. Liu, R. Deng, K.-K. R. Choo, and Y. Yang, "Privacy-preserving outsourced support vector machine design for secure drug discovery," *IEEE Transactions on Cloud Computing*, 2018.

[18] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *2007 IEEE symposium on security and privacy (SP'07)*. IEEE, 2007, pp. 321–334.

[19] B. Wang, M. Li, and H. Wang, "Geometric range search on encrypted spatial data," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 4, pp. 704–719, 2016.

[20] D. Bogdanov, S. Laur, and J. Willemson, "Sharemind: A framework for fast privacy-preserving computations," in *European Symposium on Research in Computer Security*. Springer, 2008, pp. 192–206.

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

[22] C. Lu, J. Shi, and J. Jia, "Abnormal event detection at 150 fps in matlab," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 2720–2727.

[23] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *International conference on the theory and applications of cryptographic techniques*, 1999, pp. 223–238.

[24] L. Zhang, T. Jung, K. Liu, X.-Y. Li, X. Ding, J. Gu, and Y. Liu, "Pic: Enable large-scale privacy preserving content-based image search on cloud," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 11, pp. 3258–3271, 2017.

[25] C. Piciarelli, C. Micheloni, and G. L. Foresti, "Trajectory-based anomalous event detection," *IEEE Transactions on Circuits and Systems for video Technology*, vol. 18, no. 11, pp. 1544–1554, 2008.

[26] Y. Yuan, Y. Feng, and X. Lu, "Structured dictionary learning for abnormal event detection in crowded scenes," *Pattern Recognition*, vol. 73, pp. 99–110, 2018.

[27] W. Liu, W. Luo, D. Lian, and S. Gao, "Future frame prediction for anomaly detection–a new baseline," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6536–6545.

[28] M. Sabokrou, M. Fayyaz, M. Fathy, Z. Moayed, and R. Klette, "Deep-anomaly: Fully convolutional neural network for fast anomaly detection in crowded scenes," *Computer Vision and Image Understanding*, vol. 172, pp. 88–97, 2018.

[29] T. Wang, M. Qiao, Z. Lin, C. Li, H. Snoussi, Z. Liu, and C. Choi, "Generative neural networks for anomaly detection in crowded scenes," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 5, pp. 1390–1399, 2019.

[30] X. Hu, Y. Huang, X. Gao, L. Luo, and Q. Duan, "Squirrel-cage local binary pattern and its application in video anomaly detection," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 4, pp. 1007–1022, 2019.

[31] J. T. Zhou, J. Du, H. Zhu, X. Peng, Y. Liu, and R. S. M. Goh, "Anomalynet: An anomaly detection network for video surveillance," *IEEE Transactions on Information Forensics and Security*, 2019.

[32] D. Xu, Y. Yan, E. Ricci, and N. Sebe, "Detecting anomalous events in videos by learning deep representations of appearance and motion," *Computer Vision and Image Understanding*, vol. 156, pp. 117–127, 2017.

[33] O. P. Popoola and K. Wang, "Video-based abnormal human behavior recognitionła review," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 865–878, 2012.

[34] S. Zhou, W. Shen, D. Zeng, and Z. Zhang, "Unusual event detection in crowded scenes by trajectory analysis," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 1300–1304.

[35] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[36] W. Sultani, C. Chen, and M. Shah, "Real-world anomaly detection in surveillance videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6479–6488.

[37] R. Agrawal and R. Srikant, "Privacy-preserving data mining," in *ACM Sigmod Record*, vol. 29, no. 2. ACM, 2000, pp. 439–450.

[38] C. Dwork, "Differential privacy," *Encyclopedia of Cryptography and Security*, pp. 338–340, 2011.

[39] A. C.-C. Yao, "How to generate and exchange secrets," in *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*. IEEE, 1986, pp. 162–167.

[40] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 19–38.

[41] B. D. Rouhani, M. S. Riazi, and F. Koushanfar, "Deepsecure: Scalable provably-secure deep learning," in *Proceedings of the 55th Annual Design Automation Conference*. ACM, 2018, p. 2.

[42] A. Saleem, A. Khan, F. Shahid, M. M. Alam, and M. K. Khan, "Recent advancements in garbled computing: How far have we come towards achieving secure, efficient and reusable garbled circuits," *Journal of Network and Computer Applications*, vol. 108, pp. 1–19, 2018.

[43] H. Chabanne, A. de Wargny, J. Milgram, C. Morel, and E. Prouff, "Privacy-preserving classification on deep neural network." *IACR Cryptology ePrint Archive*, vol. 2017, p. 35, 2017.

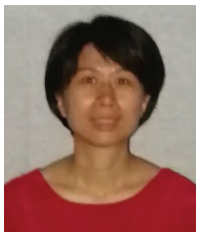
[44] E. Hesamifard, H. Takabi, and M. Ghasemi, "Cryptodl: Deep neural networks over encrypted data," *arXiv preprint arXiv:1711.05189*, 2017.

[45] Q. Zhang, L. T. Yang, and Z. Chen, "Privacy preserving deep computation model on cloud for big data feature learning," *IEEE Transactions on Computers*, vol. 65, no. 5, pp. 1351–1362, 2016.
[46] P. Li, J. Li, Z. Huang, T. Li, C.-Z. Gao, S.-M. Yiu, and K. Chen, "Multi-key privacy-preserving deep learning in cloud computing," *Future Generation Computer Systems*, vol. 74, pp. 76–85, 2017.
[47] M. Upmanyu, A. M. Namboodiri, K. Srinathan, and C. Jawahar, "Efficient privacy preserving video surveillance," in *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 1639–1646.
[48] J. Guo, P. Zheng, and J. Huang, "An efficient motion detection and tracking scheme for encrypted surveillance videos," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 13, no. 4, p. 61, 2017.
[49] X. Ma, B. Zhu, T. Zhang, S. Cao, H. Jin, and D. Zou, "Efficient privacy-preserving motion detection for hevc compressed video in cloud video surveillance," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2018, pp. 813–818.
[50] J. Guo, P. Zheng, and J. Huang, "Efficient privacy-preserving anomaly detection and localization in bitstream video," *IEEE Transactions on Circuits and Systems for Video Technology*, 2019.
[51] R. Sridharan, R. R. Maiti, and N. O. Tippenhauer, "Wadac: Privacy-preserving anomaly detection and attack classification on wireless traffic," in *Proceedings of the 11th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, 2018, pp. 51–62.
[52] H. A. Ringberg and J. Rexford, *Privacy-preserving collaborative anomaly detection*. Citeseer, 2009.
[53] H. Arai, K. Emura, and T. Hayashi, "A framework of privacy preserving anomaly detection: providing traceability without big brother," in *Proceedings of the 2017 on Workshop on Privacy in the Electronic Society*, 2017, pp. 111–122.

Hang Cheng received his BS and MS degrees in applied mathematics from Fuzhou University, Fuzhou, China, in 2002 and 2005, respectively, and PhD in signal and information processing with Shanghai University, Shanghai, China, in 2016. He is an associate professor in the Department of Information and Computational Science, College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China. He is a research scholar in the School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore. His current research interests include multimedia security, image processing, cryptography, and information hiding.

Ximeng Liu (M'16) received the B.E. degree with the Department of Electronic Engineering from Xidian University, Xi'an, China, in 2010 and Ph.D. degree with the Department of Telecommunication Engineering from Xidian University, Xi'an, China in 2015. He is currently a postdoctoral fellow with the Department of Information System, Singapore Management University, Singapore. And he is also a professor in the College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China. His research interests include applied cryptography and big data security. He is a member of the IEEE.

Huaxiong Wang received the PhD degree in mathematics from the University of Haifa, Israel, in 1996 and the PhD degree in computer science from the University of Wollongong, Australia, in 2001. He joined Nanyang Technological University in 2006 and is currently an associate professor in the Division of Mathematical Sciences. He is also an honorary fellow at Macquarie University, Australia. His research interests include cryptography, information security, coding theory, combinatorics, and theoretical computer science. He has been on the editorial board of three international journals: Designs, Codes and Cryptography (2006-2011), the Journal of Communications (JCM), and Journal of Communications and Networks. He was the program cochair of Ninth Australasian Conference on Information Security and Privacy (ACISP 04) in 2004 and Fourth International Conference on Cryptology and Network Security (CANS 05) in 2005, and has served in the program committee for more than 70 international conferences. He received the inaugural Award of Best Research Contribution from the Computer Science Association of Australasia in 2004.



Yan Fang received her BS and MS degrees in information and computation science from Fuzhou University, Fuzhou, China, in 2003 and 2006, respectively. She is a lecturer in the College of Computer and Information Sciences, Fujian Agriculture and Forestry University, Fuzhou, China. Her current research interests include multimedia security, and image processing.

Meiqing Wang received her BS and MS degrees in applied mathematics from Tsinghua University, Beijing, China, in 1987 and 1989, respectively, and PhD in Department of Computing, Xi'an Jiaotong University, China, in 2002. Now, she is a professor in the Department of Information and Computational Science, College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China. Her current research interests include computing science, image processing, and computational finance.

Xiaopeng Zhao received the B.S. degree in computer science and technology from Northeastern University at Qinhuangdao in 2014. He is currently a Ph.D. Candidate in the School of Computer Science and Software Engineering, Dept. of Cryptography and Cyber Security at East China Normal University. His research interests include public-key cryptography, computational number theory and algorithm.