SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

September 2026, Vol. 69, Iss. 9, 192503:1–192503:15
https://doi.org/10.1007/s11432-025-4878-3

Dynamically quantum attack detection for quantum key distribution based on deep representations

Minjie LIU, Ye CHEN, Xiaodong FAN, Jinquan HUANG,
Tonglin MU, Junran GUO & Shihai SUN*

School of Electronics and Communication Engineering, Sun Yat-sen University, Shenzhen 518107, China

Received 14 July 2025/Revised 25 November 2025/Accepted 17 March 2026/Published online 13 August 2026

Abstract Quantum key distribution (QKD) has attracted more attention and has been developed rapidly due to its unconditional security. However, various quantum attacks, exploiting imperfections in practical devices, might be performed by Eve to steal the key. Thus, detecting the existence of Eve and the type of quantum attacks in real time (or online) becomes an important task for practical QKD systems. In this paper, we propose a dynamic evolutionary attack detection scheme for practical QKD, in which, with the online sampled system parameters, Eve's attack feature could be vectorized by a well-designed embedding model and dynamically self-update to an attack feature vector database. Then, our method not only has high detection accuracy for known quantum attacks but also has the ability to detect and classify unseen quantum attack strategies without retraining the embedding model. Finally, taking the continuous variable (CV) QKD as an example, we demonstrate that the detection accuracy of 100% is achieved for four types of known attacks and over 99% for two types of unseen quantum attack strategies. Thus, our work significantly improves the generalization capability of quantum attack detection and promotes the practical development of QKD.

Keywords QKD, quantum attacks, dynamic evolutionary detection scheme, embedding model, attack feature vector database, unseen quantum attack strategies

Citation Liu M J, Chen Y, Fan X D, et al. Dynamically quantum attack detection for quantum key distribution based on deep representations. Sci China Inf Sci, 2026, 69(9): 192503, https://doi.org/10.1007/s11432-025-4878-3

1 Introduction

Quantum key distribution (QKD) [1–4], one of the most important technologies in quantum information science, enables unconditional secret key exchange between two remote parties (Alice and Bob). Based on the fundamental principles of quantum mechanics, QKD is rigorously proven to be unconditionally secure at the physical level, even if the eavesdropper (Eve) has unlimited computing resources. Since the first protocol was proposed, various QKD protocols have been proposed to improve their performance in different situations [5–10]. Although unconditional security of QKD has been widely studied, the practical QKD system suffers from imperfections in optical and electronic components, which leave potential loopholes for Eve to execute quantum hacking attacks. In fact, various quantum hacking attacks have been discovered [11–13], and countermeasures have also been studied to guarantee the practical security of QKD [13–18]. In all these countermeasures, online (or real-time) detection and classification of Eve's attack by tracking system parameters is an important and challenging task, especially due to the unavoidable noise of detection devices.

To overcome this problem, machine learning-based detection schemes [19–25] have been introduced. Refs. [19,20, 23] focus on training various artificial intelligence models capable of identifying known attack types. However, these models are ineffective against newly emerging, unseen attack strategies unless the datasets are updated and the networks retrained, thereby limiting their generalization capability in practical QKD systems. In [21], a GAN-based model was used, through adversarial training between generators and discriminators, which captures latent data patterns to detect both known and unseen attack strategies. Nevertheless, due to the lack of training data for specific unseen attack strategies, this method cannot effectively classify the specific attack type when identifying multiple unseen attack strategies. To mitigate this issue, a GNN-based attack detection scheme was proposed [25]. By employing a graph structure to model the datasets and incorporating the unsupervised HBOS anomaly detection

* Corresponding author (email: sunshh8@mail.sysu.edu.cn)

algorithm [26], this approach can detect multiple unseen attack strategies. However, the detection accuracy for these unseen attack strategies remains limited to 85% [25].

To address the challenge of accurately detecting both known and unseen attack strategies in practical QKD systems, this paper proposes a dynamic evolutionary attack detection scheme (DEADS) designed to detect real-time quantum attacks. Specifically, a deep representation learning embedding model is designed to vectorize the quantum attack feature vectors, and different quantum attacks (both known and unseen attack strategies) are classified by performing similarity matching with a dynamically updated attack feature vector database (AFVD). Thus, our method not only effectively improves the detection accuracy for known attacks but also has the ability to detect unseen attack strategies with high accuracy. To evaluate its performance, we apply the proposed scheme to the continuous variable (CV) QKD [27–29], which encodes a secret key in the quadrature components of coherent quantum states and shows potential advantages in improving the secure key rate and achieving a cost-effective system implementation. The experimental results demonstrate that our method has good performance in the detection of both known and unseen quantum attack strategies, reaching 100% detection accuracy for four known attack types and over 99% accuracy for two unseen attack strategies. Although the attack detection analysis in this work focuses on CV-QKD, our method is independent of the QKD protocol and is applicable to both CV and DV-QKD.

More importantly, by using our method to the CV-QKD [27–29], which encodes a secret key in the quadrature components of coherent quantum states and shows potential advantages in improving the secure key rate and achieving a cost-effective system implementation, we demonstrate that our method has good performance in the detection of both known and unseen quantum attack strategies, reaching 100% detection accuracy against four known attack types and over 99% accuracy against two unseen attack strategies. Here, we remark that although the attack detection for CV-QKD is analyzed in this paper, our method is independent of the QKD protocol and valid for both CV and DV-QKD.

This paper is organized as follows. In Section 2, we first review the CV-QKD system and perform modeling of different attack types to derive the attack characteristic parameters. In Section 3, we provide a detailed description of our proposed method, DEADS, including the representation of the attack vector, the training of the representation model, the dynamic construction of the AFVD, and the whole detection process. Section 4 presents experimental details and performance of DEADS. In Section 5, we discuss the impact of adversarial samples and countermeasures. Finally, we give a brief conclusion and outlook in Section 6.

2 CV-QKD protocol and attack feature

In this section, we first introduce the commonly used Gaussian modulated coherent state (GMCS) CV-QKD system [30]. Subsequently, we establish a mathematical model to characterize the typical quantum attacks and distill parameters that could describe the features of quantum attacks.

2.1 GMCS CV-QKD

Figure 1 illustrates a typical schematic of the GMCS CV-QKD system. Alice generates a pulse-coherent light with a laser diode (LD). The pulse light is divided into two parts by a beam splitter (BS), one weak signal as quantum light (noted as S) and a strong signal as local oscillator (noted as LO) light. For quantum signals, a variable optical attenuator (VOA) and an amplitude modulator (AM) are used to regulate the variance. An AM and a phase modulator (PM) are employed to prepare the encoded quantum state with random quadrature values x_A or p_A, which follow a Gaussian distribution with mean value zero and variance $V_x = V_A N_0$. Here, V_A is the modulation variance and N_0 is the vacuum shot noise unit variance. After polarization multiplexing with LO pulses by a polarization beam splitter (PBS), the signal pulses are transmitted to Bob through a quantum channel.

When Bob receives the pulses, he performs homodyne detection (HD) to measure the quadrature value (x_B or p_B). The mean value $\bar{y}$ and variance V_y for Bob's measured data can be written as [20]

$$\bar{y} = 0, \quad V_y = \eta T(V_x + \xi) + N_0 + V_{\mathrm{el}}, \tag{1}$$

where $T = 10^{-\alpha L/10}$ is the quantum channel transmission rate, η is the efficiency of the homodyne detector, $\xi = \varepsilon N_0$ is technical excess noise, N_0 is the shot noise and $V_{\mathrm{el}} = v_{\mathrm{el}} N_0$ is the electronic noise of the detector.

2.2 Attack modeling and feature extraction

Here, we focus on five typical quantum attack scenarios for the practical GMCS CV-QKD system: the LO intensity attack [31], calibration attack [32], saturation attack [33], and two different strategies implementing wavelength

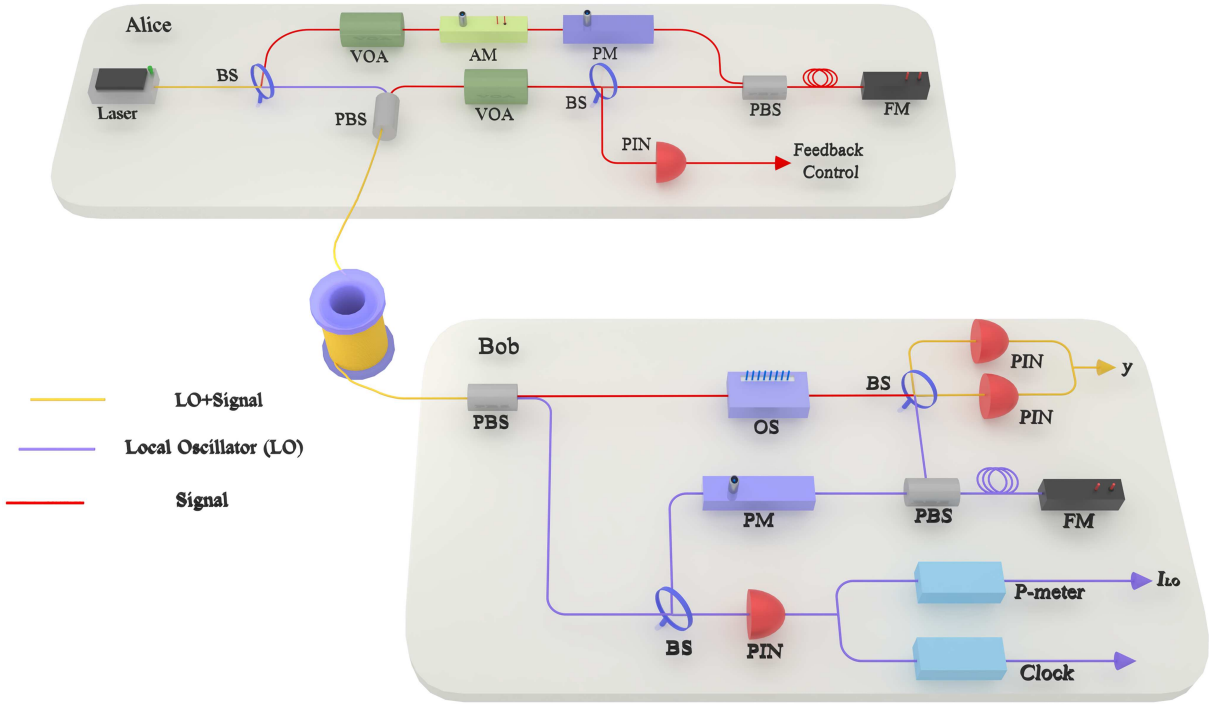

Figure 1 (Color online) Schematic diagram of a GMCS CV-QKD system. FM: faraday mirror; PIN: PIN photodiode; OS: optical switch. The red solid line represents the signal (LO) path; the yellow (blue) solid line represents the LO + signal path.

Table 1 Relationship between quantum attacks and feature parameters. The symbol ✓ (or "–") represents that the attack will (or will not) impact the feature parameters.

Attack	Feature parameters			
	$\bar{y}$	V_y	I_{LO}	N_0
LO intensity attack	–	✓	✓	✓
Calibration attack	–	✓	–	✓
Saturation attack	✓	✓	–	–
Wavelength attack-strategy-A	–	✓	✓	–
Wavelength attack-strategy-B	–	✓	–	✓

attack [34,35], one strategy manipulates the LO signal intensity (referred to as wavelength attack-strategy-A) [35], while the other is combined with the calibration attack (referred to as wavelength attack-strategy-B) [35]. With the analytical models for these attacks (see Appendix A for details), we express Bob's feature parameters, which include the intensity of the LO I_{LO}, and N_0, $\bar{y}$, V_y. All the parameters can be measured in the experiment by Bob. The relationship between these feature parameters and different quantum attacks is summarized in Table 1.

Assuming Bob receives N pulses, denoted as $\mathbf{y} = \{y_1, y_2, \ldots, y_n\}$, we randomly divide these pulses into M blocks. Then we can extract a feature vector for each block, which is given by

$$\mathbf{b} = \{\bar{y}, V_y, I_{\mathrm{LO}}, N_0\}. \tag{2}$$

Then M feature vectors $\mathbf{B}_M = \{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_M\}$ can be distilled, which could fully characterize the received pulses and detect potential attacks.

3 Dynamic evolution attack detection scheme

In this section, we introduce our dynamic evolution attack detection scheme (DEADS). In contrast to existing attack detection schemes [19–25], our DEADS utilizes an embedding model to represent the attack feature vector, and updates the represented feature vector into an attack feature vector database (AFVD). Then, by performing similarity matching between the detected attack vectors and AFVD, our method can accurately identify both known and unseen attack strategies. Additionally, when a novel attack is detected, its feature vector is automatically

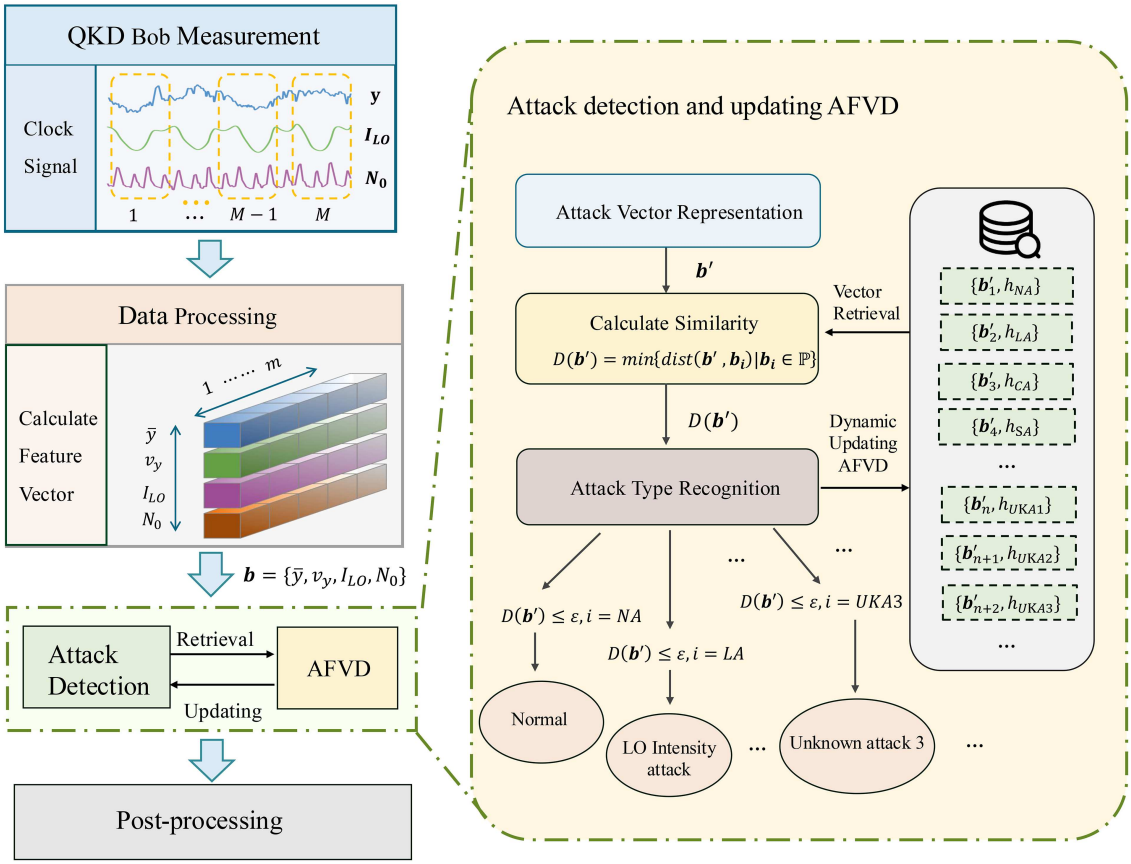

Figure 2 (Color online) The implementation process of our DEADS, including the overall framework of the DEADS and the detailed framework of attack detection and updating the AFVD module.

integrated into the AFVD in real time. This capability allows the system to evolve dynamically and detect emerging threats without the need to retrain the neural network.

3.1 Attack detection model

The proposed DEADS framework is shown in the left part of Figure 2, which consists of four components: measurement, data processing, attack detection and updating AFVD, and post-processing. The measurement part (see the Figure 1 for details) outputs the required system parameters, including LO intensity I_{LO}, the quadrature value y, and the variance of the shot noise N_0. Then the parameter vectors $\mathbf{y}_t = \{I_{\mathrm{LO}}, N_0, y\}$ at time t are obtained. After sampling N points, these collected real-time parameter vectors $\mathbf{y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_t, \ldots, \mathbf{y}_N\}$ from time t_1 to t_N are fed into the data processing module. In this module, the vectors $\mathbf{y}$ are randomly divided into M blocks, and the mean and variance are calculated for each block. This process allows for the extraction of a feature vector $\mathbf{b} = \{N_0, I_{\mathrm{LO}}, \bar{y}, V_y\}$ for each block. It is important to acknowledge that the real-time parameter sampling and data processing pose significant challenges for the practical application of our method, due to the device-specific drift, non-Gaussian noise, temporal correlations of signals, and high system repetition. However, we believe these issues can be effectively addressed. For example, the parallel real-time processing techniques developed in the quantum random number generation system [36, 37] can be introduced to our system to achieve high-speed parameter sampling and embedding vector generation.

The detailed framework for the detection and update of the attack of the AFVD module is shown in the right part of Figure 2. The attack feature vector $\mathbf{b}$ is fed into an attack vector representation module (see Subsection 3.2 for details) and the represented vector $\mathbf{b}'$ is output. This ensures that the represented attack vectors exhibit high similarity under the same attack type. To identify the type of attack, the DEADS retrieval vector from a pre-designed AFVD (noted $\mathbb{P}$) and calculate the smallest similarity distance between $\mathbf{b}'$ and the retrieved vectors in $\mathbb{P}$:

$$D(\mathbf{b}') = \min\left\{\mathrm{dist}(\mathbf{b}', \mathbf{b})_i \mid \mathbf{b}_i \in \mathbb{P}\right\}, \tag{3}$$

where the similarity distance is defined as $\mathrm{dist}(\mathbf{b}', \mathbf{b})_i$.

If the calculated similarity distance is less than a given threshold ε, the detected attack is classified as the selected attack with the smallest similarity in the attack vector dataset. Otherwise, it is classified as an unseen attack strategy. This means that the identified attack type is

$$h_b = \begin{cases} h_i, & D(\mathbf{b}') \leqslant \varepsilon, \\ h_{\mathrm{USA}}, & D(\mathbf{b}') > \varepsilon, \end{cases} \tag{4}$$

where $i \in \{\mathrm{NA}, \mathrm{LA}, \mathrm{CA}, \mathrm{SA}, \dots\}$ is the type of attack and h_{USA} represents an unseen attack strategy that is not included in $\mathbb{P}$.

During the operation of the CV-QKD system, the attack detection module detects whether the system suffers from quantum attacks or not and determines the type of attack. When the module does not find a matched feature vector in AFVD for the detected attack, it classifies the event as a new, unseen attack strategy. Then this new embedding vector $\mathbf{b}'_n$ and its label $h_{\mathrm{USA}n}$ are updated to AFVD as $\{\mathbf{b}'_n, h_{\mathrm{USA}n}\}$. Thus, the new attack vector is dynamically updated in the AFVD, and the system can detect it in real time when it occurs again.

The AFVD $\mathbb{P}$ is a data set that contains embedded vectors representing various types of attacks, which can be updated in real time during the operation of the CV-QKD system. We use discrete values to label the attack types, and these labels are the output of our attack detection system, denoted as

$$h_b \in \{h_{\mathrm{NA}}, h_{\mathrm{LA}}, h_{\mathrm{CA}}, h_{\mathrm{SA}}, \dots, h_{\mathrm{USA1}}, \dots\}, \tag{5}$$

where the labels correspond to different quantum hacking attacks, where h_{NA} represents the normal type without attack, and h_{USA1} represents the unseen attack strategy 1. During the CV-QKD system initialization process, the training data used for the training of the embedded model (see Appendix B for details) are used to initialize the AFVD, including the embedded vectors $\mathbf{b}'$ and their corresponding attack type labels h_b, which are $\mathbb{P} = \big\{\{\mathbf{b}'_1, h_{\mathrm{NA}}\}, \{\mathbf{b}'_2, h_{\mathrm{LA}}\}, \{\mathbf{b}'_3, h_{\mathrm{SA}}\}, \dots, \{\mathbf{b}'_n, h_n\}\big\}$.

It is important to note that if a quantum attack is identified during the detection phase, the data used for detection is discarded, and no key is distilled. Conversely, if no attack is detected, standard CV-QKD post-processing is performed to extract the final secret keys. This process includes parameter estimation, error correction, and privacy amplification.

3.2 Attack feature vector representation

The representation of the attack feature vector $\mathbf{b}$ is the core of our DEADS. Here, we propose an embedding-based model to effectively capture the feature vectors of different quantum attacks. Embedding techniques, which are widely utilized in natural language processing, recommendation systems, and bioinformatics [38, 39], map discrete high-dimensional data onto a continuous low-dimensional vector space. This mapping enables data processing through geometric operations (e.g., cosine similarity) within a latent space, thereby effectively capturing the semantic information, similarities, and relationships inherent in the data. In fact, an embedding function $f : \chi \to \mathbb{R}^d$ could map an input object $\mathbf{x} \in \chi$ from a discrete space to a dense vector $\mathbf{V}_{\mathbf{x}} \in \mathbb{R}^d$, where $d \ll |\chi|$.

As illustrated in Figure 3(a), the directly measured attack feature vector $\mathbf{b}$ is represented as $\mathbf{b}'$ by the embedding model. To ensure that $\mathbf{b}'$ exhibits high similarity for the same attacks, contrastive learning [40, 41] is used to train the attack embedding model by comparing the inherent similarity between samples from the same distribution (positive samples) and different distributions (negative samples). Here, the cosine similarity is adopted to quantify the similarity between two vectors, which is defined as

$$\mathrm{dist}(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|}. \tag{6}$$

The detailed model for contrastive learning is shown in Figure 3(b). First, positive and negative sample spaces from the training data are constructed, respectively. The positive sample spaces include the feature vectors of the same attacks, while the negative sample space is derived from the feature vectors of different attack scenarios. Next, a twin deep learning network is built, which processes the positive and negative samples separately to obtain positive and negative sample vectors, and the cosine similarity is calculated for each pair of vectors. The network loss function is defined as $\mathrm{dist}(\mathbf{A}, \mathbf{B})$ for the positive sample space and $1 - \mathrm{dist}(\mathbf{A}, \mathbf{B})$ for the negative sample space, respectively. This ensures that the cosine distance of the positive sample space is minimized, while that of the negative sample space is maximized. In this paper, three different networks are structured for the twin network, RNN [42], CNN [43], and Transformer [44] (see Appendix B for more details).

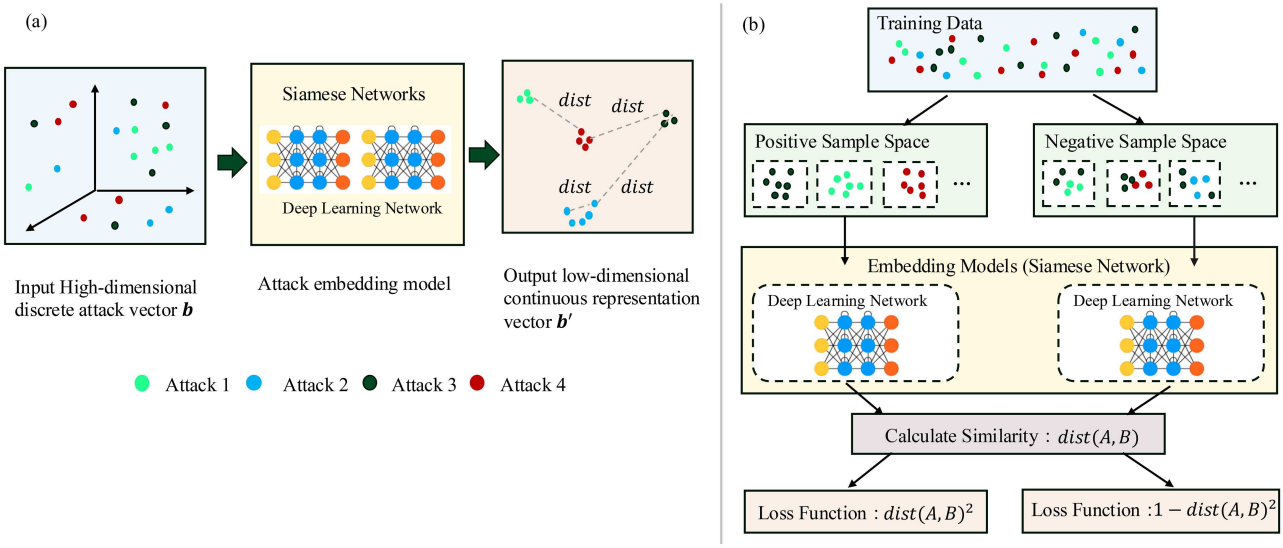

Figure 3 (Color online) Schematic diagram of attack feature vector representation. (a) Schematic diagram of attack feature vector embedding; (b) schematic diagram of training embedding model based on contrastive learning.

4 Performance analysis

4.1 Implementation details

To evaluate the performance of our method, we construct a long-distance simulation experiment with the parameters listed in Table 2. Then, based on the attack model (see Appendix A for details), we collect simulation experimental data both with and without quantum attacks. Taking the LO intensity attack as an example, the quadrature of the coherent states measured by Bob follows a Gaussian distribution with mean value 0 and variance V_y^{LOIA}, which is calculated according to the LO intensity attack model. Thus, the measured value could be generated as $\mathbf{y}^{\mathrm{LOIA}} = \{y_1, y_2, \ldots, y_n\}$. For one sample of data, we generate 25000 measured values and divide them into 25 blocks. Then the feature vector $\mathbf{b}^{\mathrm{LOIA}} = \{\bar{y}, V_y, N_0, I_{\mathrm{LO}}\}$ is calculated for each block, and 1000 simulation data are generated, with 50% used as a training set and the remaining 50% used as a test set. With the same method, we could collect data sets for no attack, calibration attack, and saturation attack. We consider wavelength attack-strategy-A and wavelength attack-strategy-B as unseen attack strategies with zero samples, in which only the testing data is required. In other words, our experimental data consist of the training set $\mathbb{Y}_{\mathrm{train}} = \{\mathbf{y}_{\mathrm{train}}^{\mathrm{NA}}, \mathbf{y}_{\mathrm{train}}^{\mathrm{LOIA}}, \mathbf{y}_{\mathrm{train}}^{\mathrm{CA}}, \mathbf{y}_{\mathrm{train}}^{\mathrm{SA}}\}$ and the test set $\mathbb{Y}_{\mathrm{test}} = \{\mathbf{y}_{\mathrm{test}}^{\mathrm{NA}}, \mathbf{y}_{\mathrm{test}}^{\mathrm{LOIA}}, \mathbf{y}_{\mathrm{test}}^{\mathrm{CA}}, \mathbf{y}_{\mathrm{test}}^{\mathrm{SA}}, \mathbf{y}_{\mathrm{test}}^{\mathrm{USA1}}, \mathbf{y}_{\mathrm{test}}^{\mathrm{USA1}}\}$.

To train the embedding model, we use the training set $\mathbb{Y}_{\mathrm{train}}$ to construct training data in both positive and negative sample spaces. To construct positive sample spaces, we perform a random cyclic shift on data of the same category. Taking the LO intensity attack as an example, through a random cyclic shift, the original sample data $\mathbf{y}_{\mathrm{train}}^{\mathrm{LOIA}} = \{\mathbf{b}_1^{\mathrm{LOIA}}, \mathbf{b}_2^{\mathrm{LOIA}}, \ldots, \mathbf{b}_{500}^{\mathrm{LOIA}}\}$ is transformed into $\mathbf{y}_{\mathrm{train}'}^{\mathrm{LOIA}} = \mathbf{b}_{1'}^{\mathrm{LOIA}}, \mathbf{b}_{2'}^{\mathrm{LOIA}}, \ldots, \mathbf{b}_{500'}^{\mathrm{LOIA}}$. Then the positive sample space $\mathbf{y}_{\mathrm{positive}}^{\mathrm{LOIA}} = \{(\mathbf{b}_1^{\mathrm{LOIA}}, \mathbf{b}_{1'}^{\mathrm{LOIA}}), (\mathbf{b}_2^{\mathrm{LOIA}}, \mathbf{b}_{2'}^{\mathrm{LOIA}}), \ldots, (\mathbf{b}_{500}^{\mathrm{LOIA}}, \mathbf{b}_{500'}^{\mathrm{LOIA}})\}$ is constructed. With the same method, $\mathbb{Y}_{\mathrm{positive}} = \{\mathbf{y}_{\mathrm{positive}}^{\mathrm{NA}}, \mathbf{y}_{\mathrm{positive}}^{\mathrm{LOIA}}, \mathbf{y}_{\mathrm{positive}}^{\mathrm{CA}}, \mathbf{y}_{\mathrm{positive}}^{\mathrm{SA}}\}$ is obtained for other attacks. To construct negative sample spaces, for each type of attack sample, samples are randomly drawn from the other three different attack samples and concatenated. Also taking the LO intensity attack as an example, we randomly select 150 samples from normal sample space $\mathbf{y}_{\mathrm{train}}^{\mathrm{NA}}$, 150 samples from the CA sample space $\mathbf{y}_{\mathrm{train}}^{\mathrm{CA}}$ and 200 samples from the SA sample space $\mathbf{y}_{\mathrm{train}}^{\mathrm{SA}}$. Consequently, the negative sample spaces for the LO intensity attack is constructed as $\mathbf{y}_{\mathrm{negative}}^{\mathrm{LOIA}} = \{(\mathbf{b}_1^{\mathrm{LOIA}}, \mathbf{b}_1^{\mathrm{NA}}), \ldots, (\mathbf{b}_{150}^{\mathrm{LOIA}}, \mathbf{b}_{150}^{\mathrm{NA}}), (\mathbf{b}_{151}^{\mathrm{LOIA}}, \mathbf{b}_{151}^{\mathrm{CA}}), \ldots, (\mathbf{b}_{300}^{\mathrm{LOIA}}, \mathbf{b}_{300}^{\mathrm{CA}}), (\mathbf{b}_{301}^{\mathrm{LOIA}}, \mathbf{b}_{301}^{\mathrm{SA}}), \ldots, (\mathbf{b}_{500}^{\mathrm{LOIA}}, \mathbf{b}_{500}^{\mathrm{SA}})\}$. With the same method, the negative sample space $\mathbb{Y}_{\mathrm{negative}} = \{\mathbf{y}_{\mathrm{negative}}^{\mathrm{NA}}, \mathbf{y}_{\mathrm{negative}}^{\mathrm{LOIA}}, \mathbf{y}_{\mathrm{negative}}^{\mathrm{CA}}, \mathbf{y}_{\mathrm{negative}}^{\mathrm{SA}}\}$ is obtained. After obtaining $\mathbb{Y}_{\mathrm{positive}}$ and $\mathbb{Y}_{\mathrm{negative}}$, we continuously adjust the value of the similarity threshold ε to achieve the best detection performance on the training set, which is selected as the similarity threshold ε.

4.2 Performance evaluation

Detection accuracy is evaluated under no attack, three known attacks, and two unseen attack strategies. Figure 4 shows the detection accuracy comparison of the existing ANN scheme [20], GAN-based scheme [21], and GNN-based scheme [25], and our DEADS scheme using the same experimental dataset. The results show that the existing ANN

Table 2 System parameters [34].

V_A	N_0	ε	v_{el}	η	α	I_{LO}	σ_{LO}	L
20	0.4	0.1	0.15	0.6	0.02 dB/km	10^7	1%	35 km

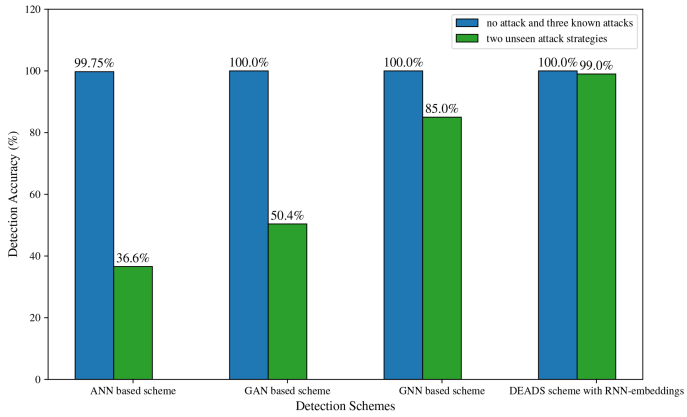

Figure 4 (Color online) Detection accuracy comparison of the existing three detection schemes and the DEADS scheme.

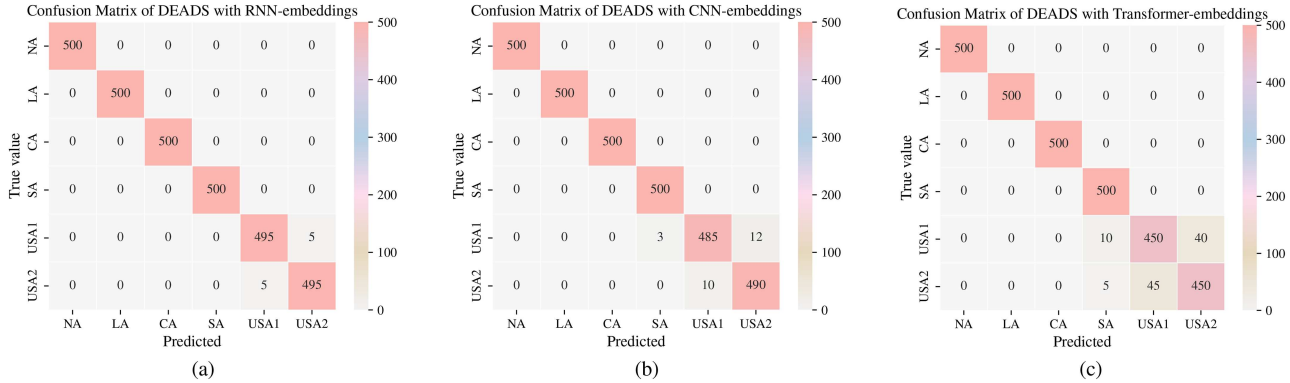

Figure 5 (Color online) Confusion matrix of DEADS. (a) DEADS with RNN-embeddings; (b) DEADS with CNN-embeddings; (c) DEADS with Transformer-embeddings.

scheme has good performance for known attacks but notably lower accuracy for the unseen attack strategies. The GAN-based scheme can detect unseen attack strategies, but the accuracy for multiple unseen attack strategies is low. The GNN-based scheme improves the accuracy of unseen attack strategies to 85%. However, our DEADS scheme achieves high accuracy for both known and unseen attack strategies.

Figure 5 shows the confusion matrix of our DEADS with three different embedding models: RNN-embeddings, CNN-embeddings, and Transformer-embeddings. The results demonstrate that the DEADS scheme achieves robust performance against all known and unseen attack strategies regardless of the embedding model used. Notably, the detection accuracy reached 99% for all three variants (Figures 5(a)–(c)).

Figure 6 shows the average accuracy of our DEAD scheme with different training epochs. For all three embedding models, the average accuracy for the no-attack and three known attacks reaches 100%, with the CNN embedding model converging the fastest. And for the detection of two unseen attack strategies, the RNN embedding model achieves the highest detection accuracy up to 99%, while 98% for the CNN embedding model, and 90% for the transformer embedding model. The RNN embedding model also converges fastest, whereas the transformer embedding model converges the slowest due to its higher complexity. Table 3 lists the precision, recall, F1-score, FNR, and FPR for attacks using the DEADS scheme with the RNN embedding model. Figure 7 shows the ROC curves and AUC scores.

4.3 Robustness and secrecy analysis

To explore the robustness of the DEADS scheme under different channel conditions, we selected the best-performing RNN-embeddings as the embedding models. Figure 8 shows the average accuracy in terms of the transmission

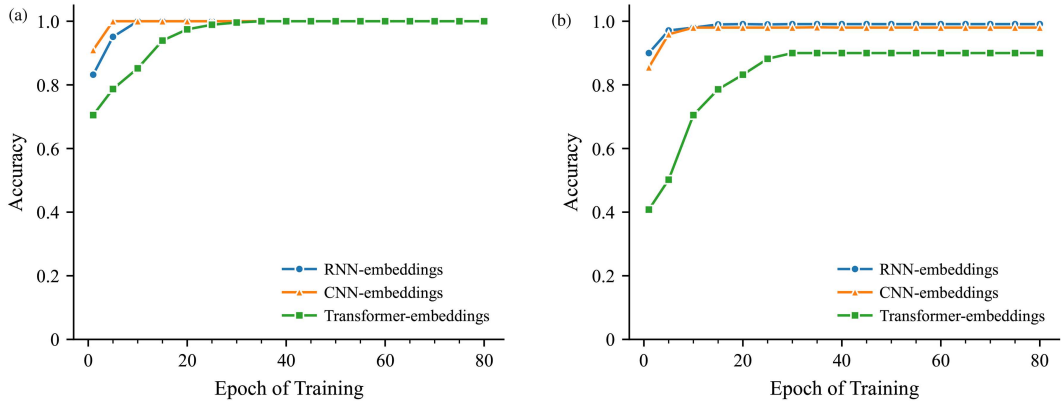

Figure 6 (Color online) The accuracy evaluation result of three different embedding models for DEADS with sufficient training data. (a) The accuracy evaluation for three known attacks; (b) the accuracy evaluation for two unseen attack strategies.

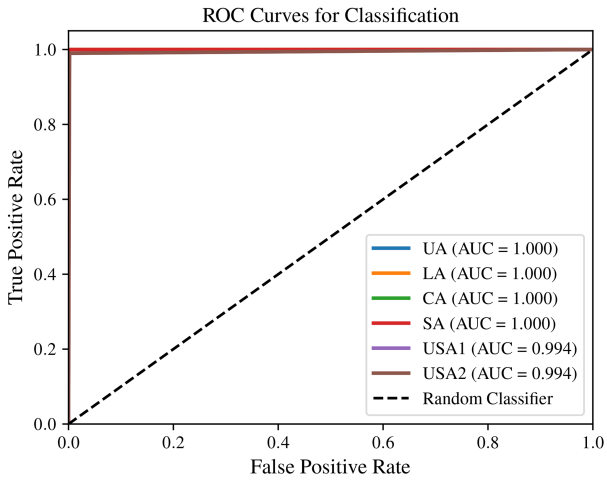

Figure 7 (Color online) ROC curves for classification of DEADS with RNN-embeddings.

Table 3 Classification report of DEADS with RNN-embeddings (L = 35 km).

	Precision	Recall	F1-score	FNR	FPR
NA	1.00	1.00	1.00	0.00	0.00
LA	1.00	1.00	1.00	0.00	0.00
CA	1.00	1.00	1.00	0.00	0.00
SA	1.00	1.00	1.00	0.00	0.00
USA1	1.00	0.99	0.995	0.01	0.002
USA2	1.00	0.99	0.995	0.01	0.002

Table 4 Classification report of DEADS with RNN-embeddings (L = 100 km).

	Precision	Recall	F1-score	FNR	FPR
NA	0.951	0.978	0.964	0.022	0.01
LA	0.978	0.98	0.979	0.02	0.004
CA	0.984	0.98	0.982	0.02	0.003
SA	0.996	0.976	0.986	0.024	0.0008
USA1	0.988	0.986	0.987	0.014	0.002
USA2	0.984	0.98	0.982	0.02	0.003

distance L. When the transmission distance L reaches 80 km, the accuracy decreases by 2%. This is because, as the transmission distance L increases, the distribution areas of different attack feature vectors become similar. This affects the detection performance of DEADS. Table 4 shows the detailed classification report of DEADS with RNN-embeddings under the transmission distance of 100 km.

Table 5 Classification report of DEADS with embedded risk preferences of RNN-embeddings (L = 100 km).

	Precision	Recall	F1-score	FNR	FPR
NA	0.984	0.976	0.98	0.024	0.003
LA	0.97	0.978	0.974	0.022	0.006
CA	0.976	0.984	0.98	0.016	0.005
SA	0.993	0.978	0.986	0.022	0.001
USA1	0.978	0.984	0.981	0.016	0.004
USA2	0.976	0.978	0.977	0.022	0.005

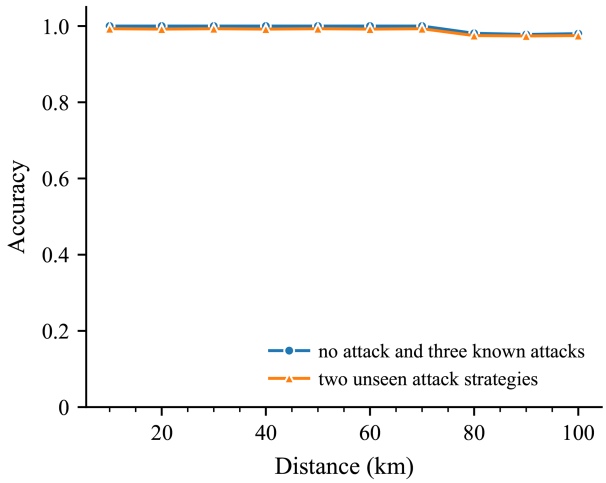

Figure 8 (Color online) The accuracy evaluation result of DEADS with RNN-embeddings in terms of the transmission distance L.

When the detector performance is imperfect, the trade-off between false-positive rate (FPR) and false-negative rate (FNR) of attack is worth discussing. If the FPR of attack is too high, the system will misjudge non-attacks, leading to excessive discarding of secure keys, increasing key generation costs, and reducing efficiency. If the FNR is too high, the system will fail to identify real attacks, allowing eavesdroppers to obtain keys and compromise security. Security is the most important aspect of QKD, which is more important than communication efficiency. The ideal balance is to minimize key loss while ensuring security. Specifically, the cost of FPR of the no attack (NA) is far greater than that of FNR of the NA. We can embed risk preferences when training the embedding model, for example, by adding a higher weight penalty to the FP samples of NA in the loss function, which forces the model to be more inclined to detect attacks. Table 5 presents the classification report for DEADS using RNN-embeddings with these embedded risk preferences. Compared with Table 4, the FPR of NA decreased from 0.01 to 0.003, and the FNR of NA only increased from 0.022 to 0.024. This optimization strategy prioritizes security without significantly sacrificing overall accuracy.

5 Adversarial examples analysis

It is important to note that, in our DEADS scheme, a stability assumption is that the training samples and testing samples approximately follow the same distribution. When low-quality and erroneous samples or even maliciously constructed abnormal samples are introduced, it may lead to misclassification. This assumption is also called adversarial examples in the AI field [45, 46]. And it is also possible to perform the adversarial examples attacks [47, 48]. The attackers generate adversarial examples by adding some tiny and subtle noise into the raw samples to interfere with the prediction stage of machine learning models, then cause misclassification. To address the vulnerability of the adversarial examples, two types of strategies [49–53] are commonly used.

The first strategy is the reactive method, which detects adversarial examples after the establishment of machine learning models. This strategy includes adversarial detection, input reconstruction, and network verification. In adversarial detection, a binary classifier is specially redesigned to detect attacks. In the input reconstruction, the process of input reconstruction is designed to purify adversarial examples by removing perturbations, thereby ensuring that the purified inputs do not cause model misclassification. The network verification can promise that, within a small perturbation, there are no existing adversarial examples to cause the neural network to misclassify.

The second strategy involves enhancing model robustness, which can be regarded as a proactive strategy. This

strategy includes adversarial training and a robust classifier. For adversarial training, training with adversarial examples is one of the countermeasures that makes neural networks more robust. Meanwhile, the robust architectures of deep neural networks are designed to prevent adversarial examples.

Although adversarial examples will affect the accuracy of our DEADS scheme, several countermeasures can be effectively employed to mitigate them. In fact, Yan et al. [22] proposed a defense scheme with artificial key fingerprints for CVQKD systems. It first embeds artificial key fingerprints into quadrature values, then exploits changes of fingerprints in the communication process to probe whether the samples have been manipulated. The results show that it is possible to monitor sample changes in real time with high precision and remain robust against adversarial example attacks. But how to construct the fingerprints for our DEADS is still unclear, especially when the unseen quantum attack strategies without training in the network are taken into consideration. We will discuss this open question in our further work. We also should note that the practical noise from the QKD system or Eve's attack might affect the validation of the adversarial examples analysis; thus, sampling data should be carefully checked in the practical application of our method. For example, the real-time data processing [36, 37] and noise-filtering [54] technologies can be used in post-processing to resist the adversarial perturbations.

6 Conclusion and outlook

Detecting quantum attacks is a crucial and challenging task for ensuring the practical security of QKD with imperfect devices, particularly regarding new and unseen attack strategies. Current methods based on machine learning have good performance in identifying known attacks but are ineffective for unseen attack strategies. In this paper, inspired by the current embedding model, we propose a dynamic evolution quantum hacking attack detection scheme for QKD, in which Bob samples the system parameter in real time, and vectorizes the features of quantum hacking through a deep representation embedding model. Then, different attacks, including both known and unseen strategies, can be classified by matching the similarity between the detected feature vector and a well-designed attack feature vector database. Using our method with CV-QKD, we demonstrate that our method performs well for both known and unseen attack strategies. Notably, the attack feature vector database can be self-updated without retraining the network when a new attack is detected, thus improving the generalization capabilities of the quantum attacks detection scheme for CV-QKD.

It should be noted that the two unseen attack strategies considered in this paper correspond to parameterized variants of the wavelength attack rather than two fundamentally new physical attack mechanisms. Whether the proposed method can detect quantum attacks based on entirely different physical mechanisms remains an open question. Addressing this issue is essential for further evaluating the generalization capability of the proposed scheme and will be investigated in our future work. Although CV-QKD is used here as a representative example, the proposed method can, in principle, also be applied to detect quantum attacks in DV-QKD systems, such as the laser seeding attack [55], time shift attack [56], and phase remapping attack [57], provided that the corresponding attack feature vectors and datasets are available. However, to make the embedding model applicable to DV-QKD systems, an additional step is required. Specifically, a mathematical model must first be established to characterize these typical quantum attacks using a finite set of measurable parameters. This requirement is not unique to our method but also applies to other AI-based approaches for quantum attack detection. Developing such models remains a significant challenge and deserves further investigation.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant No. 62171485) and Shenzhen Science and Technology Program (Grant No. JCYJ20220818102014029).

References

1 Gisin N, Ribordy G, Tittel W, et al. Quantum cryptography. Rev Mod Phys, 2002, 74: 145–195
2 Ekert A K. Quantum cryptography based on Bell's theorem. Phys Rev Lett, 1991, 67: 661–663
3 Pirandola S, Andersen U L, Banchi L, et al. Advances in quantum cryptography. Adv Opt Photon, 2020, 12: 1012–1236
4 Bennett C H, Brassard G. Quantum cryptography: public key distribution and coin tossing. Theor Comput Sci, 2014, 560: 7–11
5 Scarani V, Acín A, Ribordy G, et al. Quantum cryptography protocols robust against photon number splitting attacks for weak laser pulse implementations. Phys Rev Lett, 2004, 92: 057901
6 Cerf N J, Lévy M, Assche G V. Quantum distribution of Gaussian keys using squeezed states. Phys Rev A, 2001, 63: 052311
7 Lo H K, Curty M, Qi B. Measurement-device-independent quantum key distribution. Phys Rev Lett, 2012, 108: 130503
8 Tian Z, Xie Z, Wang R, et al. Experimental demonstration of improved reference-frame-independent quantum key distribution over 175km. Opt Express, 2024, 32: 22460–22469
9 Lucamarini M, Yuan Z L, Dynes J F, et al. Overcoming the rate-distance limit of quantum key distribution without quantum repeaters. Nature, 2018, 557: 400–403
10 Wang P, Tian Y, Li Y M. Advances in continuous variable measurement-device-independent quantum key distribution. Sci China Inf Sci, 2025, 68: 180501
11 Sun S, Huang A. A review of security evaluation of practical quantum key distribution system. Entropy, 2022, 24: 1–19
12 Xu F, Ma X, Zhang Q, et al. Secure quantum key distribution with realistic devices. Rev Mod Phys, 2020, 92: 025002

13 Jain N, Stiller B, Khan I, et al. Attacks on practical quantum key distribution systems (and how to prevent them). Contemp Phys, 2016, 57: 366–387
14 Jouguet P, Kunz-Jacques S, Diamanti E. Preventing calibration attacks on the local oscillator in continuous-variable quantum key distribution. Phys Rev A, 2013, 87: 062313
15 Yuan Z L, Dynes J F, Shields A J. Avoiding the blinding attack in QKD. Nat Photon, 2010, 4: 800–801
16 Ma X C, Sun S H, Jiang M S, et al. Enhancement of the security of a practical continuous-variable quantum-key-distribution system by manipulating the intensity of the local oscillator. Phys Rev A, 2014, 89: 032310
17 Liu W, Peng J, Huang P, et al. Monitoring of continuous-variable quantum key distribution system in real environment. Opt Express, 2017, 25: 19429–19443
18 Wang T, Huang P, Zhou Y, et al. Practical performance of real-time shot-noise measurement in continuous-variable quantum key distribution. Quantum Inf Process, 2018, 17: 1–16
19 Mao Y, Wang Y, Huang W, et al. Hidden-Markov-model-based calibration-attack recognition for continuous-variable quantum key distribution. Phys Rev A, 2020, 101: 062320
20 Mao Y, Huang W, Zhong H, et al. Detecting quantum attacks: a machine learning based defense strategy for practical continuous-variable quantum key distribution. New J Phys, 2020, 22: 083073
21 Luo H, Zhang L, Qin H, et al. Beyond universal attack detection for continuous-variable quantum key distribution via deep learning. Phys Rev A, 2022, 105: 042411
22 Yan Y, Huang D, Yin P, et al. Artificial key fingerprints for continuous-variable quantum key distribution. Phys Rev A, 2023, 108: 012601
23 Ding C, Wang S, Wang Y, et al. Machine-learning-based detection for quantum hacking attacks on continuous-variable quantum-key-distribution systems. Phys Rev A, 2023, 107: 062422
24 Xu J X, Ma X, Liu J Y, et al. Automatically identifying imperfections and attacks in practical quantum key distribution systems via machine learning. Sci China Inf Sci, 2024, 67: 202501
25 Jin D, Jiang W, Guo Y, et al. Attack detection scheme for continuous variable quantum key distribution based on a graph neural network. J Opt Soc Am B, 2024, 41: 1355–1363
26 Goldstein M, Dengel A. Histogram-based outlier score (HBOS): a fast unsupervised anomaly detection algorithm. KI 2012: Poster Demo Track, 2012, 1: 59–63
27 Braunstein S L, van Loock P. Quantum information with continuous variables. Rev Mod Phys, 2005, 77: 513–577
28 Grosshans F, Grangier P. Continuous variable quantum cryptography using coherent states. Phys Rev Lett, 2002, 88: 057902
29 Gong L H, Song H C, He C S, et al. A continuous variable quantum deterministic key distribution based on two-mode squeezed states. Phys Scr, 2014, 89: 035101
30 Grosshans F, Van Assche G, Wenger J, et al. Quantum key distribution using Gaussian-modulated coherent states. Nature, 2003, 421: 238–241
31 Ma X C, Sun S H, Jiang M S, et al. Local oscillator fluctuation opens a loophole for Eve in practical continuous-variable quantum-key-distribution systems. Phys Rev A, 2013, 88: 022339
32 Jouguet P, Kunz-Jacques S, Leverrier A, et al. Experimental demonstration of long-distance continuous-variable quantum key distribution. Nat Photon, 2013, 7: 378–381
33 Qin H, Kumar R, Alléaume R. Quantum hacking: saturation attack on practical continuous-variable quantum key distribution. Phys Rev A, 2016, 94: 012325
34 García-Patrón R, Cerf N J. Unconditional optimality of Gaussian attacks against continuous-variable quantum key distribution. Phys Rev Lett, 2006, 97: 190503
35 Huang J Z, Kunz-Jacques S, Jouguet P, et al. Quantum hacking on quantum key distribution using homodyne detection. Phys Rev A, 2014, 89: 032304
36 Guo X, Cheng C, Wu M, et al. Parallel real-time quantum random number generator. Opt Lett, 2019, 44: 5566–5569
37 Guo X, Lin J, Song Z, et al. High-entropy and parallel quantum random number generation on the strength of chaos amplification. Opt Express, 2025, 33: 41544–41557
38 Mikolov T, Chen K, Corrado G, et al. Efficient estimation of word representations in vector space. 2013. ArXiv: 1301.3781
39 Mikolov T, Sutskever I, Chen K, et al. Distributed representations of words and phrases and their compositionality. In: Proceedings of the 27th Conference on Neural Information Processing Systems, Nevada, 2013. 3111–3119
40 Hu H, Wang X, Zhang Y, et al. A comprehensive survey on contrastive learning. Neurocomputing, 2024, 610: 128645
41 Khosla P, Teterwak P, Wang C, et al. Supervised contrastive learning. In: Proceedings of the 34th Conference on Neural Information Processing Systems, Vancouver, 2020. 18661–18673
42 Al-Selwi S M, Hassan M F, Abdulkadir S J, et al. RNN-LSTM: from applications to modeling techniques and beyond-systematic review. J King Saud Univ Comput Inf Sci, 2024, 36: 102068
43 Yin W, Kann K, Yu M, et al. Comparative study of CNN and RNN for natural language processing. 2017. ArXiv: 1702.01923
44 Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need. In: Proceedings of the 31th Conference on Neural Information Processing Systems, Long Beach, 2017. 5998–6008
45 Szegedy C, Zaremba W, Sutskever I, et al. Intriguing properties of neural networks. 2013. ArXiv: 1312.6199
46 Goodfellow I J, Shlens J, Szegedy C. Explaining and harnessing adversarial examples. 2014. ArXiv: 1412.6572
47 Yuan X, He P, Zhu Q, et al. Adversarial examples: attacks and defenses for deep learning. IEEE Trans Neural Netw Learn Syst, 2019, 30: 2805–2824
48 Zhang J, Li C. Adversarial examples: opportunities and challenges. IEEE Trans Neural Netw Learn Syst, 2020, 31: 2578–2593
49 Papernot N, McDaniel P, Wu X, et al. Distillation as a defense to adversarial perturbations against deep neural networks. In: Proceedings of 2016 IEEE Symposium on Security and Privacy, San Jose, 2016. 582–597
50 Huang R, Xu B, Schuurmans D, et al. Learning with a strong adversary. 2015. ArXiv: 1511.03034
51 Metzen J H, Genewein T, Fischer V, et al. On detecting adversarial perturbations. 2017. ArXiv: 1702.04267
52 Hendrik Metzen J, Chaithanya Kumar M, Brox T, et al. Universal adversarial perturbations against semantic image segmentation. In: Proceedings of the IEEE International Conference on Computer Vision, Venice, 2017
53 Katz G, Barrett C, Dill D L, et al. Reluplex: an efficient SMT solver for verifying deep neural networks. In: Computer Aided Verification. Cham: Springer International Publishing, 2017. 97–117
54 Lin F, Ge W, Song Z, et al. Seed renewable parallel and real-time Toeplitz post-processing for QRNG. J Lightwave Technol, 2024, 42: 8606–8615
55 Huang A, Navarrete Á, Sun S H, et al. Laser-seeding attack in quantum key distribution. Phys Rev Appl, 2019, 12: 064043
56 Qi B, Fung C H F, Lo H K, et al. Time-shift attack in practical quantum cryptosystems. Quantum Info Comput, 2007, 7: 73–82
57 Xu F, Qi B, Lo H K. Experimental demonstration of phase-remapping attack in a practical quantum key distribution system. New J Phys, 2010, 12: 113026

Appendix A Mathematical analysis model of attacks

To analyze the impact of quantum hacking attacks on practical CV-QKD systems, we performed mathematical modeling and analysis of several typical quantum hacking attacks. We focus on LO intensity attack, calibration attack, saturation attack, and wavelength

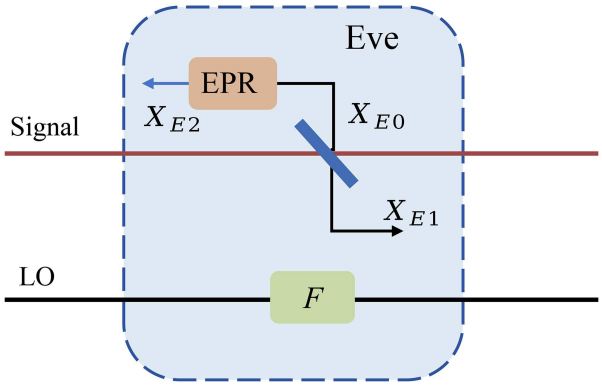

Figure A1 (Color online) The LO intensity attack schematic diagram [31]. Eve introduces a Gaussian collective attack in the signal pulse to steal the secret key. The LO signal strength is reduced by manipulating F.

attack with strategy A (referred to as wavelength-attack-strategy-A), wavelength attack with strategy B (referred to as wavelength-attack-strategy-B).

In the LO intensity attack [31], the attack schematic diagram is shown in Figure A1. Eve intercepts the signal pulse by introducing Gaussian collective attacks [34], ξ_{Gau} is the additional noise introduced by Gaussian collective attacks, ξ_{Gau} can be expressed as

$$\xi_{\mathrm{Gau}} = \frac{(1-\eta T)(N-1)}{\eta T N_0}, \tag{A1}$$

where N is the variance of Eve's EPR states, N can be expressed as

$$N = \frac{1-k\eta T}{k(1-\eta T)}. \tag{A2}$$

In order to hide the attack, Eve manipulates the LO signal intensity by using an intensity attenuator with an attenuation coefficient $k \in (0,1)$ to reduce the excess noise estimated by Alice and Bob. The variance of the quadrature value V_y^{LOIA} and the intensity of the LO under LO Intensity Attack can be expressed as

$$V_y^{\mathrm{LOIA}} = k\left[\eta T(V_A N_0 + \xi + \xi_{\mathrm{Gau}}) + N_0 + V_{\mathrm{el}}\right], \tag{A3}$$

$$I_{\mathrm{LO}}^{\mathrm{LOIA}} = k I_{\mathrm{LO}}. \tag{A4}$$

The shot noise N_0^{LOIA} under LO intensity attack can be expressed as

$$N_0^{\mathrm{LOIA}} = k N_0. \tag{A5}$$

In the calibration attack [32], Eve implemented the partial intercept-resend (PIR) attack[1)] to signal pulses and introduced a phase-independent attenuator to modify the local oscillator pulse shape. The excess noise introduced by the calibration attack can be expressed as

$$\frac{\xi_{\mathrm{CA}}}{N_0} = \frac{N_0^{\mathrm{CA}}}{N_0}\left[\frac{\xi_{\mathrm{PIR}}}{N_0^{\mathrm{CA}}} + \frac{1}{\eta T}\left(1 - \frac{N_0}{N_0^{\mathrm{CA}}}\right)\right], \tag{A6}$$

where the excess noise introduced by PIR and shot noise after calibration attack can be expressed as

$$\xi_{\mathrm{PIR}} = \xi + 2\gamma N_0, \tag{A7}$$

$$N_0^{\mathrm{CA}} = \frac{N_0}{1+(2\gamma+0.1)\eta T}, \tag{A8}$$

where N_0 is the shot noise before the calibration attack. In order to hide the attack, the excess noise estimated by Alice and Bob should be close to zero, so

$$\frac{N_0}{N_0^{\mathrm{CA}}} = 1 + 2\eta T, \tag{A9}$$

therefore, variance of the quadrature value V_y^{LOIA} under calibration attack can be expressed as

$$V_y^{\mathrm{CA}} = \eta T(V_A N_0^{\mathrm{CA}} + \varepsilon N_0^{\mathrm{CA}} + 2N_0^{\mathrm{CA}}) + N_0^{\mathrm{CA}} + v_{\mathrm{el}} N_0^{\mathrm{CA}}. \tag{A10}$$

1) Lodewyck J, Debuisschert T, García-Patrón R, et al. Experimental implementation of non-Gaussian attacks on a continuous-variable quantum-key-distribution system. Phys Rev Lett, 2007, 98: 030503.

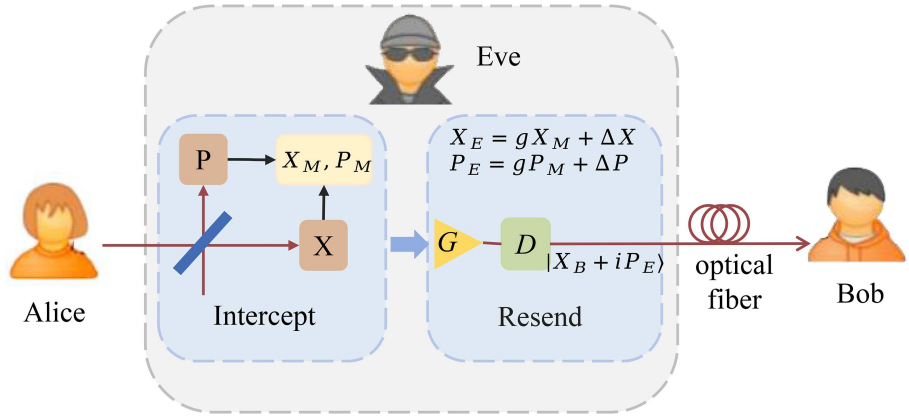

Figure A2 (Color online) The saturation attack schematic diagram [33].

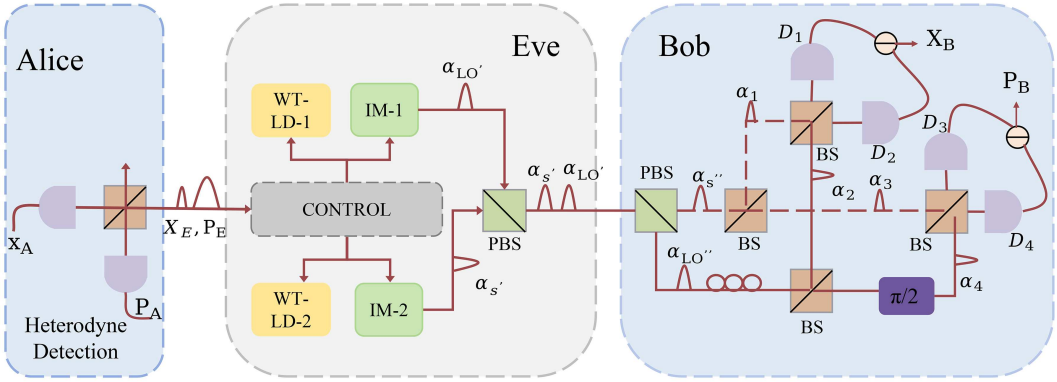

Figure A3 (Color online) The wavelength attack schematic diagram [35]. The WT-LD and IM are used in producing fake coherent states with the specific wavelength and amplitude set by the controller.

In the saturation attack [33], the attack schematic diagram is shown in Figure A2. Eve uses heterodyne detection to implement the intercept-resend attack (IRA). During the quantum state preparation in the resend step, Eve displaces the quadrature coherent states with a value Δ to make the detector work in the saturation region. When the detector is working in the linear region, the variance of the quadrature value V_y^{Lin} can be expressed as

$$V_y^{\mathrm{Lin}} = \eta T(V_A N_0 + \xi + 2N_0) + N_0 + V_{\mathrm{el}}. \tag{A11}$$

When the detector is working in the saturation region, the excess noise estimated by Alice and Bob will be reduced, which achieves the purpose of the hidden attack. The measurement mean and variances under the saturation attack can be expressed as

$$\bar{y}^{\mathrm{SA}} = \alpha - \sqrt{\frac{V_y^{\mathrm{Lin}}}{2\pi}} B + \frac{\alpha - \Delta}{2} + \frac{\alpha - \Delta}{2} A, \tag{A12}$$

$$V_y^{\mathrm{SA}} = V_y^{\mathrm{Lin}} \left(\frac{1 + A}{2} - \frac{B^2}{2\pi} \right) - (\alpha - \Delta) \sqrt{\frac{V_y^{\mathrm{Lin}}}{2\pi}} AB + \frac{(\alpha - \Delta)^2 (1 - A^2)}{4}, \tag{A13}$$

where

$$A = \mathrm{erf}\left(\frac{\alpha - \Delta}{\sqrt{2V_y^{\mathrm{Lin}}}} \right), \tag{A14}$$

$$B = \mathrm{e}^{-\frac{(\alpha - \Delta)^2}{2V_y^{\mathrm{Lin}}}}. \tag{A15}$$

The attack schematic diagram of wavelength attack [34, 35] is shown in Figure A3. Because of an imperfection in the wavelength-dependent coupling ratio of the fiber beam splitter, Eve can vary the transmittance of the fiber beam splitter by inserting lights with different wavelengths when implementing IRA. There are two strategies to implement the wavelength attack [34], one strategy is to cooperate with manipulating the LO signal intensity, called wavelength attack-strategy-A, and the other strategy is to cooperate with the calibration attack called wavelength attack-strategy-B.

Table B1 Architecture of the RNN-embeddings model.

Layer type	Output shape	Parameters
Input	[32, 4]	–
RNN network	[32, 64]	2080
LSTM	[32, 25, 8]	768
Linear	[32, 25, 4]	36
Normalization	[32, 25, 4]	8
Linear	[32, 25, 4]	20
Linear	[32, 64]	6464
RNN network	[32, 64]	2080
LSTM	[32, 25, 8]	768
Linear	[32, 25, 4]	36
Normalization	[32, 25, 4]	8
Linear	[32, 25, 4]	20
Linear	[32, 64]	6464
Total		9376

In wavelength attack-strategy-A, Eve manipulates the LO signal intensity and inserts lights with different wavelengths when implementing IRA. The measurements can be expressed as follows:

$$V_y^{WA-A} = r_i\eta T(V_A N_0 + \xi + 2N_0) + V_{\mathrm{el}} + N_0^{WA-A}, \tag{A16}$$

where

$$N_0^{WA-A} = \frac{N_0}{\sigma} + (1 - r_i^2)D_{WA-A}^2 + (35.81 + 35.47r_i^2)D_{WA-A}, \tag{A17}$$

$$I_{\mathrm{LO}}^{\mathrm{HA1}} = \frac{I_{\mathrm{LO}}}{\sigma}, \tag{A18}$$

and σ is the attenuation coefficient, I_s and I_{LO} are the resend pulse of signal and LO, λ_s and λ_{LO} are wavelengths of signal and local light. D_{WA-A} is a coefficient obtained with the intensities I_s and wavelength λ_s for signal (and I_{LO}, λ_{LO} for for local light). r_i is the attenuation ratio of amplitude modulation in measuring shot noise.

In wavelength attack-strategy-B, Eve controls the slope of the homodyne detection response by calibrating the trigger time. γ is the effective slope coefficient, which can be biased by modifying homodyne detection to calibrate the trigger time parameters. The measurements can be expressed as follows:

$$V_y^{WA-B} = \gamma\eta T(V_A N_0 + \xi + 2N_0) + V_{\mathrm{el}} + N_0^{WA-B}, \tag{A19}$$

$$N_0^{WA-B} = \gamma N_0 + (1 - r_1 r_2)D_{WA-B}^2 + (35.81 - 35.47r_1 r_2)D_{WA-B}. \tag{A20}$$

Appendix B Embedding model architecture and training details

Inspired by contrastive learning [40, 41] and the typical network architecture of deep learning: recurrent neural network (RNN) [42], convolutional neural network (CNN) [43], and Transformer [44], we propose three embedding models based on the Siamese networks using RNN, CNN, and Transformer, respectively.

In the RNN-embedding model, as illustrated in Table B1, a Siamese network was constructed using two RNN-based subnetworks with shared weights. Each subnetwork incorporates a bidirectional LSTM layer that captures temporal dependencies in both forward and backward directions of the attack feature vectors. Following this, a fully connected layer is employed for feature dimensionality reduction, integrated with residual connections, and a normalization layer is applied to accelerate convergence. After flattening the features, subsequent fully connected layers perform global feature extraction, mapping the original 4-dimensional attack feature vectors into compact 64-dimensional attack embedding vectors.

In the CNN-embedding model, as illustrated in Table B2, a Siamese network was constructed using two CNN-based subnetworks. Within each subnetwork, the first convolutional layer captures local fluctuations, while the second convolutional layer integrates global trends through residual connections. The fully connected layer flattens the convolutional output of the attack features and maps them to a high-dimensional space, resulting in 64-dimensional attack embedding vectors.

In the Transformer-embedding model, as illustrated in Table B3, a Siamese network was constructed using two Transformer-based structures. In the Transformer-based network, a linear layer serves as the embedding layer to adjust the vector dimensions. Positional encoding is then applied to add positional information to the input, followed by dropout for regularization and normalization. The input is processed through each transformer encoder layer, with layer normalization applied afterward. A residual connection is added by summing the output of the transformer layer with the original input. After flattening, the final output is produced through a fully connected linear layer, resulting in a 128-dimensional attack embedding vector. Unlike RNN-embedding and CNN-embedding models, in order to capture global relationships, the Transformer should expand the feature space through its self-attention mechanism, which requires higher dimensions to accommodate these expanded features. We try to find a high-performing model architecture as an embedding model for our DEADS. Thus, according to the results Figures 5 and 6, we focus on RNN-embedding, but do not optimize the Transformer-embedding with a lower dimension.

Table B2 Architecture of the CNN-embeddings model.

Layer type	Output shape	Parameters
Input	[32, 4]	–
CNN network	[32, 64]	–
Convolution (1D)	[32, 10, 4]	760
ReLU	[32, 10, 4]	20
Convolution (1D)	[32, 25, 4]	775
Batch normalization (1D)	[32, 25, 4]	50
LeakyReLU	[32, 25, 4]	–
Linear	[32, 64]	6464
CNN network	[32, 64]	–
Convolution (1D)	[32, 10, 4]	760
ReLU	[32, 10, 4]	20
Convolution (1D)	[32, 25, 4]	775
Batch normalization (1D)	[32, 25, 4]	50
LeakyReLU	[32, 25, 4]	–
Linear	[32, 64]	6464
Total		8069

Table B3 Architecture of the Transformer-embeddings model.

Layer type	Output shape	Parameters
Input	[32, 4]	–
Transformer network	[32, 128]	–
CLinear	[32, 25, 4]	20
Dropout	[32, 25, 4]	–
Normalization	[32, 25, 4]	8
Transformer encoder layer	[32, 25, 4]	136
Normalization	[32, 25, 4]	8
Transformer encoder layer	[32, 64]	126
Normalization	[32, 25, 4]	8
Linear	[32, 128]	12928
Transformer network	[32, 128]	–
CLinear	[32, 25, 4]	20
Dropout	[32, 25, 4]	–
Normalization	[32, 25, 4]	8
Transformer encoder layer	[32, 25, 4]	136
Normalization	[32, 25, 4]	8
Transformer encoder layer	[32, 64]	126
Normalization	[32, 25, 4]	8
Linear	[32, 128]	12928
Total		13244