Security and Communication Networks

Security and Defense of Cyber-Physical Systems

Lead Guest Editor: Yuanyuan Huang
Guest Editors: Malu Zhang and Haomiao Yang

Security and Defense of Cyber-Physical Systems

Security and Communication Networks

Security and Defense of Cyber-Physical Systems

Lead Guest Editor: Yuanyuan Huang
Guest Editors: Malu Zhang and Haomiao Yang

Contents

Hindawi
Security and Communication Networks
Volume 2023, Article ID 6234030, 9 pages
https://doi.org/10.1155/2023/6234030

Research Article

Twitter Bots in Cyber-Physical-Social Systems: Detection and Estimation Based on the SEIR Model

Weisha Zhang,[1] **Jiazhong Lu**,[2] **Yulin Liu**,[2] **and Xiaojun Liu**[2]

[1]*School of Foreign Languages, University of Electronic Science and Technology of China, Chengdu 611731, China*
[2]*Advanced Cryptography and System Security Key Laboratory of Sichuan Province, School of Cybersecurity, Chengdu University of Information Technology, Chengdu 610225, China*

Correspondence should be addressed to Jiazhong Lu; ljz@cuit.edu.cn

Received 1 September 2022; Revised 11 November 2022; Accepted 16 April 2023; Published 8 May 2023

Academic Editor: Malu Zhang

Bots are now part of the social media landscape, and thus, a threat to cyber-physical-social systems (CPSSs). A better understanding of their characteristic behaviors and estimation of their impact on public opinion could help improve the algorithms to identify bots and help develop strategies to reduce their influence. The cosine function-based algorithm is able to compare the similarity between tweets and restore the course of information circulation. Combined with malicious features of an account, our method could effectively detect bots. We implement SEIR model to compute tweets with the hashtag #Huawei 5G and divide the trend propagation into the following four phases: formation, fermentation, explosion, and decay of trend. Sentiment analysis revealed the change of emotion and opinion among normal users in different stages and the manipulation attempt of bots behind it. Experiment results show that bots have very limited relation to users' stance in whole. In early phase bots could affect those who are neutral. The influence of bots declines in later stage. Polarized views can hardly be changed.

1. Introduction

For the past few years, devices or systems have been transformed into smart connected ones, which are widely named as Internet of Things (IoT) and cyber-physical systems (CPSs). Integrating social networks to CPSs results in a new paradigm called cyber-physical-social systems (CPSSs). CPSSs include the human-to-device communications and device-to-device communications and create continuous interaction of human-device relationships [1]. In recent years, with the development of artificial intelligence technology, machine algorithms can automatically generate fake news stories, spam, misinformation, and other deceptive content. Bots are used to distribute misleading information over social media networks. They are used as a tool for public opinion manipulation. By selling anxiety, creating conflicts, spreading rumors, and slander, bots have been used to manipulate online discussion and shape public life. Their activities are also associated with spamming and harassment. Bots are regarded as initiator of current challenges towards CPPSs, such as privacy concerns, ethical issues, safety, and security.

There are the following three types of bot accounts: hecklers, hacker bots, and honeypots. Hecklers attack the opponent or promote own interest by using analogous talking points. Hacker bots compromise the social media accounts of celebrities to post disinformation or gain access to private communications. Honeypot accounts are used to contrive friendships between normal users and bots, building trust through direct messaging or email conversations and then trick users into making payments into fraudsters' accounts [2]. Thus, malicious social media bots are a cyber threat to information security and to our society.

Machine-learning algorithms could help identify bots. Their judgement bases mainly on the behavior characteristics of accounts. To improve the efficiency of bot detection we need a better understanding of bots' activity in social media and their actual impact on public opinion. Therefore, we choose Twitter, one of the most popular social media in the world, as our sample. Latest Pew Research study shows that Twitter is by far the most used social platform among journalists [3]. While much of the reporting is influenced by

tweets opinion, researchers estimate that from 9% to 15% of the active Twitter accounts are bots [4]. Automated accounts can have an outsized impact, as the 500 most active bots shared 22% of the total links on Twitter [5]. Our research focuses first on distinguishing between bots and normal users, second on bots' activity in debate of the hot topic #Huawei 5G and their impact on public opinion.

2. Related Work

2.1. Feature-Based Bot Detection. Compared with normal users, bots show great differences in behavior patterns, in social networks and in the way of information dissemination. Using extracted features to train bot detection classifier is the current major practice. Features refer to the following three categories: the text content, the behavior patterns, and the relationship to others. The choice of features has direct impact on classifier quality. Researchers worldwide are looking to optimize algorithms though better characteristic classification.

Content-based bot detection may be accomplished in the following two ways: (1) Through text similarity calculation. Automation controls a large number of fake accounts. That same or similar tweets or hashtags appear at the same instant in a large number is a typical act of bots. (2) Through specific content in posting like URLs. Bots often induce users to visit the target website by sending them advertisements, phishing, or pornographic information which contains URLs pointing to a same target. Content-based bot detection methods generally involve natural language processing, text similarity calculation, text sentiment tendency analysis, word segmentation, and other technologies. Andriotis and Takasu [6] present a supervised approach to detect automated accounts on Twitter using four datasets that contain users' metadata, content, and sentiment features. Kumar et al. [7] proposed a neural network ensemble of Text CNN and LSTM model with BERT embeddings to classify tweets as bot tweets or not based on the tweets' textual content. Heidari et al. [8] create new features based on textual information of online comments. The new set of sentiment features are extracted from a tweet's text and used to train bot detection models. The advantage of the content-based bot detection is that the trained model can effectively distinguish bots with known characteristics quickly. The disadvantage is that bots can evade detection by disguising URLs, changing text generation templates, and using automated tools to produce a large number of texts with different words but similar meanings.

Behavior-based bot detection depends on abnormal behavioral features such as abnormal posting time distribution, frequency, and time series. Methodologically, pattern matching, time series analysis, user behavior modeling, and statistical analysis are used. Chu et al. [9] proposed a detection system that consists of two main components, a client-side logger and a server-side classifier, to distinguish bots from humans. Dorri et al. [10] present a semisupervised collective classification technique that combines the structural information of the social graph with the information on the social behavior of users in a unified manner to detect social botnets in a Twitter-like SNS. The behavior-based bot detection is suitable for bots with typical abnormal behaviors. It has higher robustness and bots can hardly evade.

SNS-based bot detection relies on uncovering abnormal connections among accounts. Bots do not have normal social relationships like human. Their social networks distinguish in partial structure and attributes. Social Networks-based bot detection involves technologies such as network construction, social network analysis, complex network, graph mining, community detection, network presentation, and cluster analysis. Sengar et al. [11] compiled activity and profile information of users on Twitter and using NLP and supervised machine learning to achieve the objective classification. Zhang et al. [12] propose a rectified linear post-synaptic potential function for spiking neurons and a spike-timing-dependent BP-learning algorithm for DeepSNNs. Their model bases on statistical features and user bi-directional voting. The statistical features include bi-directional propagation between trust users and neighbor nodes. Social Networks-based bot detection can discover individual bot and collaborative army of bots.

2.2. Information Diffusion in the SEIR Model. The Susceptible-Infective-Removed (SIR) model is one of the most widely used models for the information diffusion research in social networks. Many researchers have devoted themselves to improving the classic SIR model in different aspects. Rui et al. [13] proposed a Susceptible-Potential-Infective-Removed (SPIR) model that analyses the diffusion process based on the discrete time according to simulation. Sang and Liao [14] proposed a novel information dissemination model in mobile social networks based on the traditional SIR model (SEIRD) to know the evolution trend of information over time. Jia et al. [15] distinguished two propagation channels of rumor spreading on social networks, proposed an improved SIR model, and established the corresponding mean-field equation. Fu et al. [16] investigated the dynamics of competitive information diffusion over a connected social network by presenting a modified SIR model and found that innovators and larger network degree can help enlarge the coverage of the information among the population but they cannot help one information to compete with the other one.

3. Data Collection and Methods

3.1. Data Sets. We crawled through Twitter API all tweets tagged with the hash #Huawei 5G from November 1, 2019 to December 31, 2019 and got 22,950 samples. The data include content features (Table 1) and account features (Table 2).

3.2. Filtering Nonoriginal Tweets by Cosine Function-Based Algorithm. Since there are a lot of repetitive contents, lexical and semantical filtering is necessary. We use the cosine similarity algorithm to analyze the similarity of contents and eliminate duplications. The cosine similarity can compare the similarity of two tweets in terms of content. Each tweet is given two attribute vectors, A and B. The remaining string

Table 1: Content features.

Field name	Description
Article	The original text of tweet posted by the user
Author	User name/login name
Time	Posting time, year/month/day/hour/minute
likenum	Number of likes on the tweet
replynum	Number of replies to this tweet
forwardnum	Number of retweets for this tweet
article_id	The URL address of the tweet
author_id	The link to author's homepage

Table 2: Account features.

Field name	Description
user_id	Personal ID generated by account registration
Name	User name
Location	Residential address selected by the user at the time of registration
createtime	Time of account creation
fansnum	Number of followers of the account
friendsnum	Number of other users that this account follows
statusnum	Number of tweets posted by the account
medianum	Number of videos and other multimedia posted by the account
favoritenum	Number of tweets collected by the account
screen_name	User-created login name
Verified	Whether the user account is authenticated, 1 means yes, 0 means no
Fpf	Followers-to-friends ratio
originalp	Tweets to all posting ratio
retweetp	Percentage of retweets, i.e. (1-originalp)
Quantity	Total number of tweets posted by this account
regtime	Registration time
isBot	Determine whether the user is a bot

similarity θ is given by the dot product and vector length as follows:

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}, \quad (1)$$

where A_i and B_i represent the components of vectors A and B. The similarity is measured by cosine of the angle. When two vectors point to the same direction and overlap, the value is 1, indicates that the contents of two tweets are identical; when it is the 90°angle, the value is 0, means that the contents are independent. When the value is greater than 0.8, it is regarded as repeated semantic duplicate content and is eliminated.

3.3. Bot Identification Based on Text Content and Account Feature. Bots are problematic because they manipulate information, spread misinformation, promote unverified information, and adversely affect public opinion. To detect bots, we first start with the behavioral characteristics to find out suspicious accounts and then combine with malicious account characteristics to identify bots.

First, excluding tweets with identical content; then, sorting tweets with cosine similarity greater than 0.6 by post time to find the earliest atomic sentence. Based on that, the later added statements are shown in timeline and the organizational development of contents is restored. Next, marking the time point when the replies increase rapidly and labeling the statement that caused this change as key sentence. Finally, using account push to find out the account which first posted the key sentence. The more the key sentence repeats, the larger its value is, and the more suspicious the account is. The flow is as shown in Figure 1.

Consider that the accounts that caused a large number of retweets could be either activity of organized bots or influential opinion leader, further detection is needed. We extract user account characteristics including all postings, original postings, video postings, fan following ratio, number of replies, number of likes, number of favorites, length of registration, and whether certified. We purpose a logistic linear regression function to perform machine learning on the extracted user features to detect bots.

3.4. Information Diffusion on Hot Topics According to the SEIR Model. The SEIR model of infectious disease dynamics is divided into four stages. Applying to information diffusion,

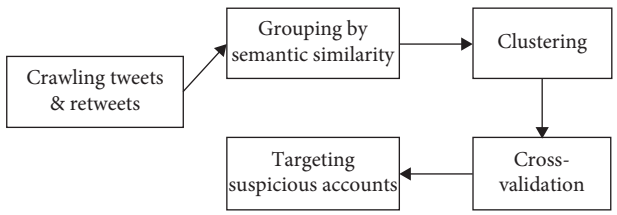


Figure 1: Flowchart of bot detection.

they are Stage S (formation period), in which the news spread on the Internet but has not triggered public attention jet.; Stage E (fermentation period), in which the public forms opinions, attitudes, and emotions about the event gradually; Stage I (outbreak period) is characterized with drastic fluctuation, large volume of information, and clear emotional tendency. Stage R (dissipation period) in which public attention decreases and the topic of heat vanishes.

Most social networks are dimensionless, a scale-free model reflects the normality better than the one based on random networks. We use the Fitness model as archetype, with m_0 bots in the network at the beginning. For each additional time interval, a new batch of bots with rank m will join in. And the present bot i could connect to the new bot. The probability of connection hangs on the rank and adaptability of the old one. The relationship is shown as follows:

$$\prod i = \frac{\eta_i k_i}{\sum_j \eta_j k_j}, \tag{2}$$

where $\prod i$ is the probability of connection, η_i is the rank, and k_i is the adaptability. Let $N(t)$ be the total number of bots at time t, equal to m_0 plus t. Although infectious disease and information do not spread exactly the same, they do share some similarities. They need connection and disseminators. The Susceptible-Exposed-Infective-Recovered-Susceptible (SEIRS) model is used to describe the information diffusion. Let $S(t)$, $E(t)$, $I(t)$, and $R(t)$ be the amounts of bots at time t and represent: S is the one who has not been contacted; E is the one who may have interest in the topic; I is the one who join the discussion; R is the one who quit the issue. Their transformation is shown in Figure 2.

$\beta, \omega, \gamma, \sigma, \alpha$ are the triggering rate of bots in different stages. The triggering rate of bots depends on the topic and on the rank of other bots in the same scale-free network. We modeled them as follows:

$$\beta_i = \beta_0 \frac{k_i}{\sum_j K_j}. \tag{3}$$

β_i is the transition probability from S to E. β_0 is the infection rate of the topic. Suppose that the total number of bots in E induced by bot i equals k_i. In this time unit, the number can be expressed as follows:

$$\sum_{t=1}^{i=E(t)} \frac{\beta_i k_i S(t)}{N(t)}. \tag{4}$$

Let β be $\sum_{t=1}^{i=E(t)} \beta_i k_i$, the model of information diffusion on scale-free social networks can be expressed as follows:

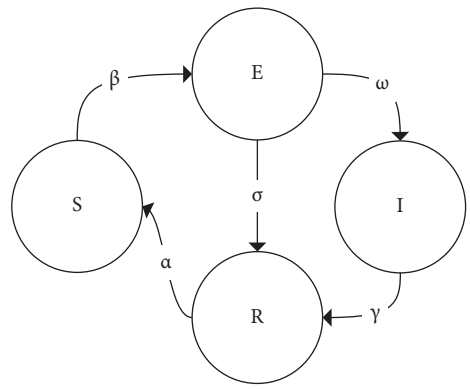


Figure 2: Four stages transition.

$$\begin{cases} \dfrac{\mathrm{dS}(t)}{\mathrm{dt}} = \alpha R(t) - \dfrac{\beta S(t) E(t)}{N(t)}, \\ \dfrac{\mathrm{dE}(t)}{\mathrm{dt}} = \dfrac{\beta S(t) E(t)}{N(t)} - (\omega + \delta) E(t), \\ \dfrac{\mathrm{dI}(t)}{\mathrm{dt}} = \omega E(t) - \gamma I(t), \\ \dfrac{\mathrm{dR}(t)}{\mathrm{dt}} = \delta E(t) + \gamma I(t) - \alpha R(t), \\ N(t) = S(t) + E(t) + I(t) + \mathrm{R}(t). \end{cases} \tag{5}$$

4. Results and Analyses

4.1. Experimental Environment. The experimental hardware environment is shown in Table 3.

4.2. Trend in the SEIR Model. We analyze the topic "Huawei 5G" on Twitter according to the SEIR model. The timeline is shown in Table 4.

We used natural language processing to analyze the four stages of tweets content into the following three tendencies: positive (blue), negative (green), and neutral (yellow). Figure 3 reflects the trend of emotional tendencies in four stages.

This shows that (1) neutral tweets were the most in all phases; (2) opinions are the most divergent in Stage I. The battle of views drives the discussion to the peak; (3) in Stage E, the participation drops slightly than the beginning, the public are looking forward further information to their own judgement. Thus, Stage E is an import time point for manipulating or guiding public opinion; (4) in Stage R, neutral netizens quit the discussion, while the dissenter stay. It indicates that social network users with a clear emotional orientation are more willing to follow the topic.

4.3. Verification of Bot Detection. We had manually mark 10,179 accounts for the dataset. Account considered as bot is marked as 1, otherwise as 0. The dataset is used for algorithm training and performance verification.

Table 3: Experimental environment.

Hardware name	Version
Operating system	Windows 11 64 bit
Processor	12th Gen Intel(R) Core (TM) i7-12700H 2.30 GHz
Memory	12th Gen Intel(R) Core (TM) i7-12700H 2.30 GHz 16g

Table 4: Time period of topic #Huawei 5G.

Stage	Time quantum
S-stage	2019.11.01 00:01:27–2019.11.1706:01:27
E-stage	2019.11.17 06:01:27–2019.12.0106:01:27
I-stage	2019.12.01 06:01:27–2019.12.2206:01:27
R-stage	2019.12.22 06:01:27–2019.12.3123:58:11

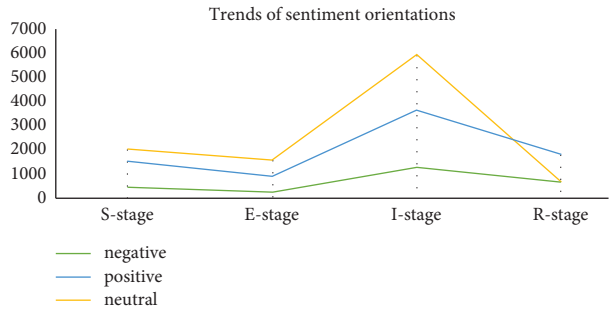


Figure 3: Trend of sentiments.

We used TP (True Positives), FN (False Negatives), FP (False Positives), and TN (True Negatives) to classify the 2000 accounts of the prediction model. TP indicates that the model correctly detected the bot; TN indicates that the model incorrectly detected the bot as normal user; FP indicates that the model incorrectly detected the normal user as bot; and FN indicates that the model incorrectly detected the bot as normal user. Accuracy tells how many times the model was correct overall. Precision is how good the model is at predicting a bot account. Recall tells how many times the model was able to detect a bot account. The formula for calculation is as shown in Table 5.

In this experiment, the values of TP, TN, FP, and FN are 347, 1637, 353, and 357, which means the accuracy is 99.2%, precision is 98.3%, and recall is 97.2%. It indicates that the bot detection model used in the experiment has high accuracy and can provide support for further experiments.

4.4. Bot Activities. First, we count the number of tweets in the whole process and visualized as shown in Figure 4.

We can see that (1) bot activities are found in all four stages; (2) among the total 22950 tweets, 19111 were from normal users, that is 83.27%; 3839 were from bots, that is 16.73%; and (3) the trend peaked at 18:01:27 on 12.05.2019. At that point 437 tweets were posted, 86% were from normal users and 14% were from bots.

The following Figures 5–8 show the bot activities at each stage:

It is to see that (1) bots were most active in the *S* and *R* stage and contributed 18.54% and 18.74% of all tweets, 2% more than the average, while normal users were more active in the E and I stage and posted 85.57% and 84.31% of all tweets. This shows a clear difference between their behaviors and (2) bots were highly involved in early discussion and promoted the formation of hot topic. When the topic of heat drops, bots will once again inrush into the discussion and reverse the trend.

4.5. Bot Influence on Public Opinion. In the period from 11.01.2019 to 12.31.2019 we get 22,950 tweets on the whole. We categorize them according to affective tendencies into three groups, red for affirmative, yellow for negative and green for neutral. The emotional tendency of normal users and bots are shown in Figure 9.

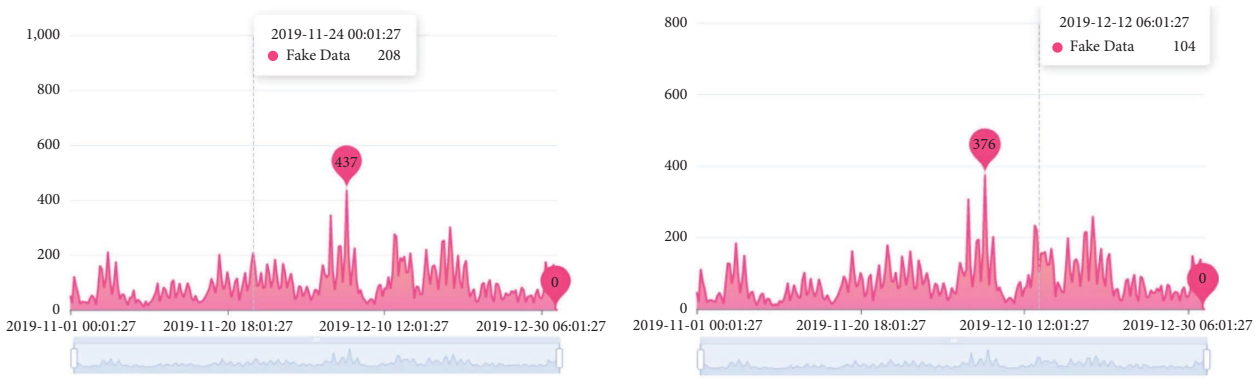
Visualizing Figure 10 with a parallel coordinate chart we get Figure 9. For the same color, dark one represents the affective tendency of normal users, while light one shows the public opinion including bots as a whole. That the two lines are almost in parallel, means that the influence of bots is minimal.

To get a better understanding of the difference of bots and human in sentiment tendencies, we show them in percentage. The light color *o* indicates normal users and the dark indicates bots, Figure 11.

We assume that bots only perform tasks and won't change mind, while normal users do. Normal users with neutral attitudes continuously decline from stage E, while the one with opposing attitudes increase, indicating that the

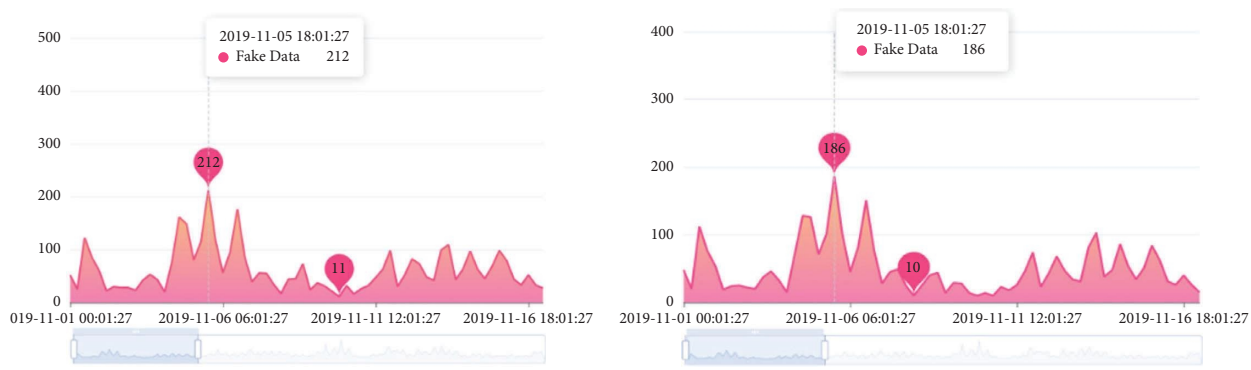
Table 5: Algorithm performance evaluation indicators.

Metric	Definition
Accuracy	(TP + TN)/ (TP + FP + TN + FN)
Precision	(TP)/ (TP + FP)
Recall	(TP)/ (TP + FN)



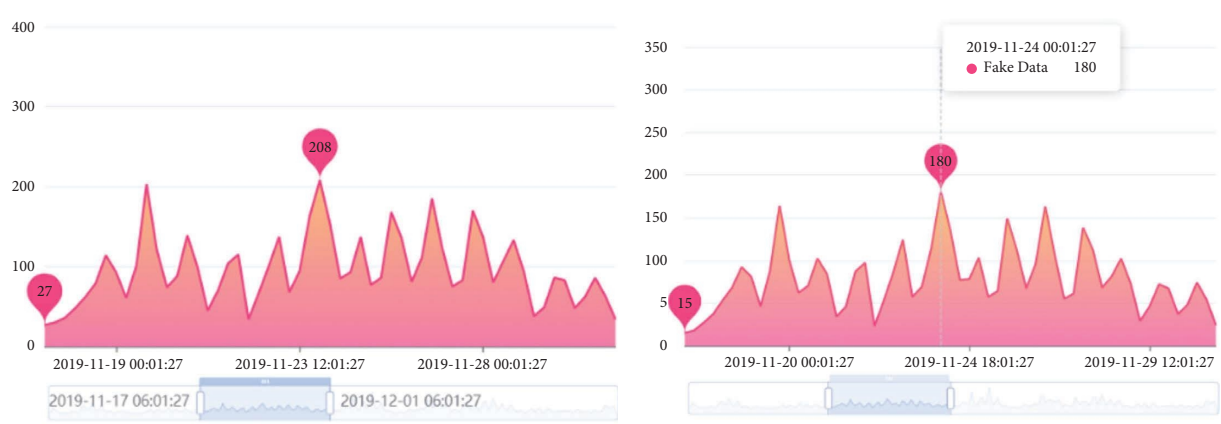
(a) (b)

Figure 4: Tweets (a) inclusive and (b) exclusive bots' postings.



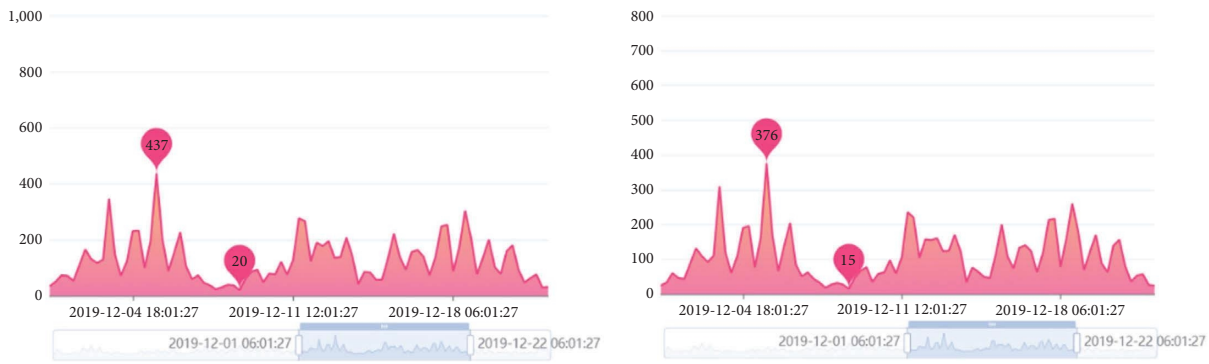
(a) (b)

Figure 5: Tweets in S-stage (a) inclusive and (b) exclusive bots' postings.



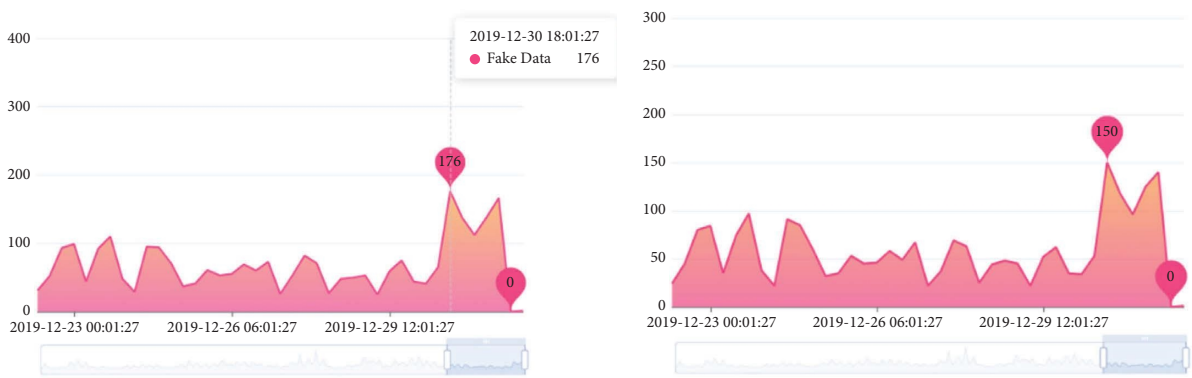
(a) (b)

Figure 6: Tweets in E-stage (a) inclusive and (b) exclusive bots' postings.



(a) (b)

Figure 7: Tweets in I-stage (a) inclusive and (b) exclusive bots' postings.



(a) (b)

Figure 8: Tweets in E-stage (a) inclusive and (b) exclusive bots' postings.

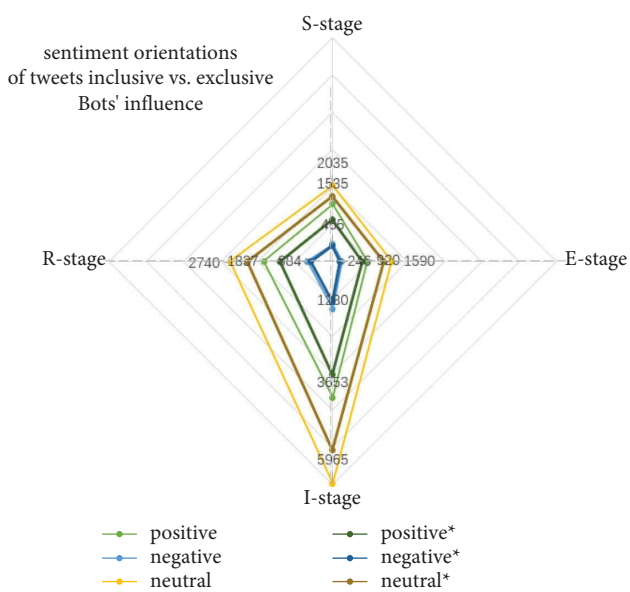


Figure 9: Sentiments inclusive vs. exclusive bots' influence.

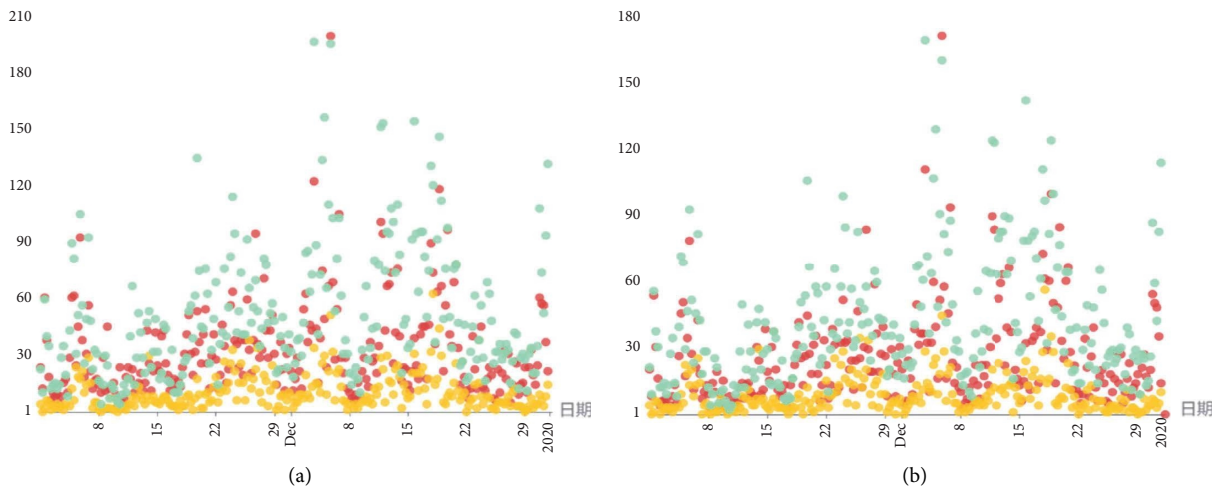


Figure 10: Tweets of different sentiment orientations (a) inclusive and (b) exclusive those posted by bots.

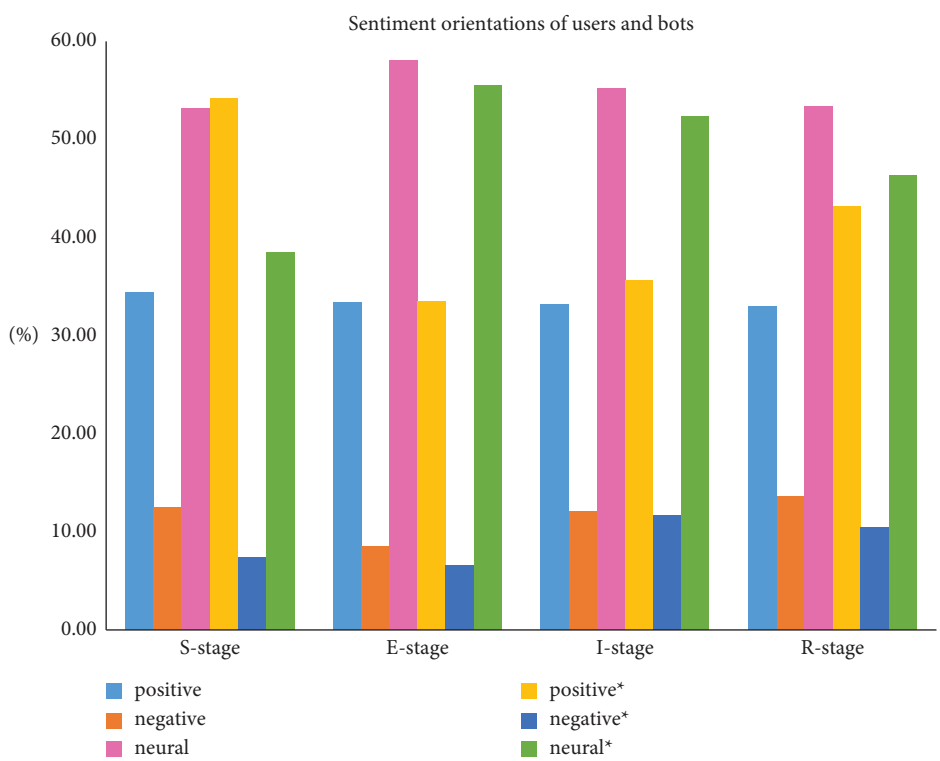


Figure 11: Sentiment orientations of users and bots.

neutrals become opponents. Reasons for the change could be: first, the approval bots reduce 19.73% from S to E stage and second and the negative bots increased significantly from E to I stage. Normal users with supportive attitude remain stable. They are less influenced by external opinion and stick to their own views.

5. Conclusions

Bots are automated accounts used to engage in social media. They are often blamed for opinion manipulation on divisive issues, because bots can spread information rapidly and amplify specific content strategically. Working in unison, they can maximize impact and give the illusion of large-scale consensus.

Our study focuses on improving bot detection techniques and evaluating the diffusion of discussions on social media by bots. First, we used a cosine function-based algorithm to judge the similarity between texts and extract content related features. In combination of malicious account features through machine learning process we are able to identify bots efficiently. Then, we applied the SEIR Model from epidemic study to trends analysis. A controversial topic experiences generally formation (S), fermentation (E),

explosion (I), and decay (R) of trend. Our question is whether and to what degree can bots influence users' perspectives. Bots have very limited on people's view if we take the trend as a whole, but in different stages their influence is diverse. In early phase bots could affect the users who take a neutral position. Bots can win the amorphous middle by setting an echo chamber around them. In opposite, the late-moving birds catch no worm, because the influence of bots declines in later stages.

Not all bots engage in public opinion manipulation. From the sentiment analysis we can see that even within the bot group, neutral ones are the majority. A possible explanation could be that those neutral tweets were posted by good bots and fake good bots. The latter post promotion, critics and general content in combination, in order to better imitate a human user and thus evade detection methods. Good bots often identify themselves clearly as bots and tweet useful information and latest news. Making good bots more influential on social media networks is a way to combat malicious automation.

Bots exist on all kinds of social media platform. Current research focuses mainly on Twitter bots. One reason for this is the unwillingness of platforms to share data on account activity. It makes researchers difficult to analyze message frequency, networks, or employ other techniques to identify bots. Some bot behavior may be universal, but there are some platform-specific characteristics which deserve more attention.

Data Availability

The data supporting the current study are available from the corresponding author upon reasonable request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was part of the project "Computational Communication Strategy for Chinese Technology Image on Twitter" (no. 20CXW016) which is supported by National Social Science Fund of China. This work was also supported by the National Natural Science Foundation of China (no. 62102049) and the Open Fund of Advanced Cryptography and System Security Key Laboratory of Sichuan Province (Grant no. SKLACSS-202110).

References

[1] R. Reine, F. H. Juwono, Z. A. Sim, and W. K. Wong, "Cyber-physical-social systems: an overview," *Smart Connected World*, pp. 25–45, 2021.

[2] J. Berger, C. Watts, and A. Weisburd, "Trolling for trump: how Russia is trying to destroy our democracy," 2016, https://warontherocks.com/2016/11/trolling-fortrump-how-russia-is-trying-to-destroy-our-democracy.

[3] G. Jeffery, M. Amy, J. Mark, and L. Jakob, "Journalists sense turmoil in their industry amid continued passion for their work. Pew research center," 2022, https://www.pewresearch.org/journalism/2022/06/14/journalists-sense-turmoil-in-their-industry-amid-continued-passion-for-their-work.

[4] O. Varol, E. Ferrara, C. A. Davis, F. Menczer, and A. Flammini, "Online human-bot interactions: detection, estimation, and characterization," 2018, https://arxiv.org/abs/1703.%2003107.

[5] S. Wojcik, S. Messing, A. Smith, L. Rainie, and P. Hitlin, "Bots in the twittersphere. Pew research center," 2018, http://www.pewinternet.org/2018/04/09/bots-in-thetwittersphere.

[6] P. Andriotis and A. Takasu, "Emotional bots: content-based spammer detection on social media," in *Proceedings of the 2018 IEEE international workshop on information forensics and security (WIFS)*, pp. 1–8, IEEE, Hong Kong, China, December 2018.

[7] S. Kumar, S. Garg, Y. Vats, and A. S. Parihar, "Content based bot detection using bot language model and BERT embeddings," in *Proceedings of the 2021 5th International Conference on Computer, Communication and Signal Processing (ICCCSP)*, pp. 285–289, IEEE, Chennai, India, May 2021.

[8] M. Heidari, H. James Jr, and O. Uzuner, "An empirical study of machine learning algorithms for social media bot detection," in *Proceedings of the 2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*, pp. 1–5, IEEE, Toronto, ON, Canada, April 2021.

[9] Z. Chu, S. Gianvecchio, and H. Wang, "Bot or human? A behavior-based online bot detection system," in *From Database to Cyber Security*, pp. 432–449, Springer, Cham, 2018.

[10] A. Dorri, M. Abadi, and M. Dadfarnia, "Socialbothunter: botnet detection in twitter-like social networking services using semi-supervised collective classification," in *Proceedings of the 2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*, pp. 496–503, IEEE, Athens, Greece, August 2018.

[11] S. S. Sengar, S. Kumar, P. Raina, and M. Mahaliyan, "Bot detection in social networks based on multilayered deep learning approach," *Sensors & Transducers*, vol. 244, no. 5, pp. 37–43, 2020.

[12] M. Zhang, J. Wang, J. Wu et al., "Rectified linear postsynaptic potential function for backpropagation in deep spiking neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 5, pp. 1947–1958, 2022.

[13] X. Rui, F. Meng, Z. Wang, G. Yuan, and C. Du, "SPIR: the potential spreaders involved SIR model for information diffusion in social networks," *Physica A: Statistical Mechanics and Its Applications*, vol. 506, pp. 254–269, 2018.

[14] C. Y. Sang and S. G. Liao, "Modeling and simulation of information dissemination model considering user's awareness behavior in mobile social networks," *Physica A: Statistical Mechanics and Its Applications*, vol. 537, Article ID 122639, 2020.

[15] P. Jia, C. Wang, G. Zhang, and J. Ma, "A rumor spreading model based on two propagation channels in social networks," *Physica A: Statistical Mechanics and Its Applications*, vol. 524, pp. 342–353, 2019.

[16] G. Fu, F. Chen, J. Liu, and J. Han, "Analysis of competitive information diffusion in a group-based population over social networks," *Physica A: Statistical Mechanics and Its Applications*, vol. 525, pp. 409–419, 2019.

Hindawi
Security and Communication Networks
Volume 2023, Article ID 6802359, 15 pages
https://doi.org/10.1155/2023/6802359

Research Article

SR2APT: A Detection and Strategic Alert Response Model against Multistage APT Attacks

Fan Shen,[1] **Levi Perigo**,[1] **and James H. Curry**[1,2]

[1]*Department of Computer Science, University of Colorado Boulder, Boulder, CO 80309, USA*
[2]*Department of Applied Mathematics, University of Colorado Boulder, Boulder, CO 80309, USA*

Correspondence should be addressed to Fan Shen; fan.shen@colorado.edu

Received 21 July 2022; Revised 9 October 2022; Accepted 11 October 2022; Published 19 April 2023

Academic Editor: Yuanyuan Huang

Advanced persistent threats are an emerging cyber threat to cyber-physical systems (CPS), especially those comprising mission-critical physical assets. However, defense against such attacks is challenging, due to their sophistication, stealthiness, and zero-day exploitation. Existing works in this area mainly focus on the detection of APT, but it might be too late or too costly to impede APT when it is detected with high confidence. Therefore, this work focuses on CPS intrusion detection and prevention against APT attacks and aims at preventing such attacks in earlier stages through a strategic response policy to imperfect APT alerts by leveraging the multistage characteristic of APT and a deep reinforcement learning formulation. A novel host-based APT detection and response model called SR2APT is proposed, which consists of a detection engine and a decision engine. The detection engine is based on graph convolutional network, which classifies a stream of system log provenance subgraphs as an APT stage or benign. Then, the detection results are transmitted to the decision engine sequentially, which is trained based on deep reinforcement learning and outputs the optimal response actions to APT alerts. Experimental results show that the GCN-based detection engine obtains 94% classification accuracy on a semisynthetic dataset of system logs and outperforms classification models based on SVM, CNN, and LSTM. The strategic alert response policy from the decision engine is compared with two baseline fixed response policies, and it achieves the best trade-off between preventing APT attacks and minimizing the impediments of mistaken active defense actions to benign activities that generate false alerts, thus obtaining the highest total rewards in the defense against APT attacks.

1. Introduction

Cyber-physical systems (CPS) integrate software components with physical processes, which provide monitoring and feedback loops to maintain and improve the performance of physical processes [1]. As the physical processes in CPS are usually mission-critical, such as the power grid or programmable logic controllers, these systems have been targeted by advanced persistent threats (APT) [2]. APT is a type of sophisticated cyberattack [3, 4], which aims at stealing valuable and possibly confidential data or causing the malfunction of the mission-critical infrastructure of the victim without being detected. Initially, the targets of APT are government or military sectors for espionage, but over recent years, the APT battlefield has extended to the IT infrastructure of high-profile companies [5], IoT networks [6], and public cloud platforms [7] for sabotage purpose or financial gains. Considering the devastating outcome of APT, any modern organizations should be aware of it and have protection mechanisms in place to defend against it, and the leveraging unique traits of sophisticated attacks like APT for accurate and agile detection is a focus of CPS intrusion detection system design [8].

Defense against APT presents an evolving challenge due to the unique characteristics of such attacks. Different from general computer viruses or worms that intend to cause widespread and indiscriminate damage, APT attacks focus on specific targets, e.g., a server storing sensitive data or a mission-critical infrastructure component, just like the programmable logic controllers (PLCs) compromised in

Stuxnet [3], therefore, attack vectors are usually well planned and customized for the target. In order to access the target without being detected, APT attackers move low and slow to maintain persistence in the victim's network, and these long-term campaigns span multiple stages, such as reconnaissance, weaponization, delivery (of malicious code), exploitation, installation (of malware), command and control communication, and action on objective [9]. Most importantly, APT is advanced as these well-resourced attackers are capable to exploit zero-day vulnerabilities. The attackers in Stuxnet, believed to be responsible for the malfunction of centrifuges in Iran's nuclear project, exploited more than four zero-day vulnerabilities [10].

Existing studies on APT detection can be categorized as either point detection, focusing on special behaviors or specific stages of APT attackers, or contextual detection combining the footprints of different APT stages. In terms of point detection, network packets and flows, especially those related to http(s) traffic, are analyzed in [11–13] to identify the command and control (C2) communication, which is commonly used by APT malware to receive instructions from or exfiltrate data to malicious servers. E-mail features are extracted to detect spear phishing e-mails used by APT attackers to deliver malicious PDF files or URLs to the victim's system [14, 15]. CPU and memory utilization, the content of the Windows Registry, and the System32 directory are also useful sources to capture the footprints of APT malware [16, 17]. Host-level system logs are analyzed in [18] to map a sequence of system logs to an APT stage. However, point detection methods may cause lots of false positives when anomalous behaviors can result from both APT and benign activities [19]. To reduce false positives, contextual detection methods correlate detection results of multiple APT stages. An attack pyramid model is proposed in [20] providing a thorough conceptual framework to link APT alerts of different stages across various planes. An attack chain model is used in [19] to describe the APT life cycle, and anomalous events from different APT stages are combined using statistical methods to assign scores to hosts, indicating the level of compromise by APT. A bottom-up approach is proposed in [21] to map low-level system logs to TTPs of APT defined by MITRE ATT & CK using predefined rules and then build a chain of TTPs based on node relations on the provenance graph and apply scoring rules to differentiate APT scenarios and benign scenarios.

This work focuses on the defense against a multistage APT attack on a host and considers the threat model in which the behavioral pattern of each APT stage can appear in both attack and benign scenarios. Therefore, it is insufficient to claim the existence of APT by the detection of a single APT stage. Existing works that tackle similar APT scenarios [19, 21] using contextual detection methods have demonstrated the feasibility to detect multiple APT stages from system logs. The goal of these works is to reduce false positives in APT detection by correlating alerts of multiple APT stages. However, there is no defender's intervention that would affect system activities during the collection of APT alerts, and it can be too late or too expensive to impede APT when it is detected with confidence. In this paper, an APT detection and alert response model called SR2APT is proposed to provide agile and strategic responses to imperfect APT alerts to secure critical hosts in CPS. On the one hand, SR2APT integrates the detection of APT stages and strategic responses to APT alerts such that a multistage APT is prohibited in earlier stages, which is more advantageous than ex post facto remedies when APT has progressed to later and more destructive stages. On the other hand, SR2APT applies a model-free deep reinforcement learning method to solve strategic alert response actions, which does not require assumptions about the strategies of the APT attacker compared with game theoretic studies. The two main components of SR2APT are described below, with more technical details in Section 4.

First, the detection engine in SR2APT is responsible for classifying a provenance subgraph, which is extracted from host-level system logs, as one stage of a multistage APT or a general benign class. A graph convolutional network (GCN) is used for the multiclassification task, which leverages useful features hidden in the structure of a graph. To the authors' knowledge, this is the first work of using GCN to analyze system logs for the detection of APT stages.

Second, the decision engine in SR2APT outputs optimal response actions to alerts from APT stages. It is trained based on deep reinforcement learning, which incorporates the defender's understanding of the threat model, interdependencies between APT stages, costs, and benefits of response actions. Although APT alerts can be triggered by both APT and benign activities in the threat model of this paper, the alert response strategy from the decision engine is able to prevent APT attacks in earlier stages and minimize the impediments to benign activities from mistaken active defense actions. Unlike existing contextual APT detection methods, SR2APT takes the defender's responses into consideration during detection, thus it enables cost-effective proactive defense against multistage APT attacks.

For evaluations, the detection engine of SR2APT is tested on a semisynthetic dataset, which includes host-level system logs about malicious behaviors of six APT stages and benign behaviors. The decision engine of SR2APT is evaluated on synthetic alert traces from both APT and benign scenarios, which are generated according to the interactions between the defender and the attacker. Experimental results show that the GCN-based detection engine achieves 94% classification accuracy and outperforms other kernels including SVM (support vector machine), one-dimensional CNN (convolutional neural network), and LSTM (long short-term memory). The strategic alert response policy from the decision engine achieves the highest total rewards in the defense against APT when compared with two baseline fixed response policies, because it optimally balances the trade-off between maximizing the number of prevented APT attacks and minimizing the impediments of mistaken active defense actions to benign activities that generate false alerts.

The remaining of this paper is outlined as follows. Section 2 reviews related works about defending against APT by the analysis of system logs. Section 3 presents the architecture of the proposed APT detection and alert response model, SR2APT. Section 4 provides technical details

of the main components of SR2APT. Section 5 shows the design and results of the proof of concept experiments. Section 6 concludes this work and suggests directions for future work.

2. Related Work

System logs record essential activities executed by users, applications, or operating systems, serving as an important source for digital forensic analysis [22]. Each entry in system logs is known as an event, including the fields defined by the system log capturing tools, such as the event time, enter process name and ID, exit process name and ID, and event type. A trace of system logs is a sequence of events in the temporal order. Signature-based methods can be used to check if an event is malicious or not. For example, a simple string matching method is applied to system logs for the detection of data exfiltration, which is one of the main goals of APT attacks [23]. In this work, malicious actions such as rename, copy, and move to sensitive files are captured by system logs, and then a policy-based method is used to enforce a set of rules to detect malicious activities and send alert notifications. However, signature-based methods can only detect known attack patterns and require that the rules are well-defined, complete, and updated. Instead of focusing on a single event, occurrence-based methods are used to extract features of system behaviors that relate to multiple events. In [24], a trace of system logs is represented by short and fixed-length contiguous sequences, and then metrics based on Hamming distance are used to decide if a trace of system logs is malicious by comparing its representation with the representation of benign traces. An anomaly detection method based on k-nearest neighbors is proposed to analyze system logs in [25]. In that paper, the occurrences of each type of system calls are collected to form the feature vector of a trace of system logs. For an unknown trace, cosine similarity is applied to find its k-nearest neighbors in the feature space, and then those neighbors' classes are used to determine its class. To consider the ordering of system calls, Subba et al. [26] used the frequencies of unique n-grams to form the feature vector of a trace of system logs, where an n-gram is defined as a contiguous system calls of length. However, when dealing with large-scale real-world system logs, the number of unique n-grams can grow exponentially, and the dimension of the frequency vector increases, thus it becomes intractable to enumerate all patterns.

To efficiently detect APT by analyzing system logs, recent works suggest using provenance graphs or information flow graphs to structure system logs and then either obtaining more meaningful sequences of system logs from provenance graphs or applying graph-based methods to analyze provenance graphs. The advantage of provenance graphs is that system calls are connected based on their causal relations rather than being temporally ordered [27], meaning that two interdependent system calls spanning a long period can be more efficiently connected by provenance graphs than raw system logs. Figure 1 shows a provenance graph built from a sample trace of system logs.

The hidden Markov model (HMM) is used to study host-level system logs in [18] for identifying multiple APT phases. In this work, a sequence of low-level system logs pertaining to an APT phase is first extracted from a provenance system; next, it is translated to a sequence of abstracted states called storylines by HMM. Then, these high-level state sequences are fed into three multiclassification models: LSTM, 1-dimensional CNN, and SVM, to predict the APT phase of a sequence of system logs. Their results show that using HMM-based storylines outperforms using the original sequence of system logs in the multiclassification models. However, HMM requires prior knowledge of definitions of hidden states, and such models are computationally expensive.

HOLMES [21] is a bottom-up framework to detect APT from system logs with the help of provenance graphs. The three layers in this model from low to high are system logs, TTPs (tactics, techniques, and procedures) defined by MITRE ATT&CK, and APT stages. First, a provenance graph is generated from system logs, and then TTPs are identified from local structures of the graph based on predefined rules that are matched by node properties, edge types, and distances between nodes. Finally, a weighted product scoring rule is applied to aggregate a chain of TTPs and assign it a threat score. Though HOLMES is able to assign distinguishable threat scores to APT and benign scenarios, since it relies on predefined rules to map system logs to TTPs, this model has high requirements for prior knowledge.

A Bayesian LSTM neural network is used to detect APT system log traces [27], in which these fixed-length traces are obtained from a host-level system provenance graph. Context information and neighborhood information are encoded for each event node in the provenance graph. The advantage of BNN is that it provides uncertainty to the predictions. This work focuses on predicted APT traces with low uncertainty and constructs the attack story for such a trace by finding its most similar trace in the training set. However, their way of constructing the attack story assumes that the APT traces in the training set are well understood, which can be difficult to guarantee as the split of the training set and testing set usually requires data shuffling and random sampling.

UNICORN [28] is an APT detection model that periodically summarizes the graph histogram by counting unique subtrees in the whole system provenance graph of a host and compares graph histograms efficiently using graph sketching. It detects anomalies by building evolutionary models for regular behaviors and using unsupervised learning to determine the state of current graph histogram in each model, if the state does not match the state transition pattern of any evolutionary models, it indicates the existence of APT. However, UNICORN can lead to high false positives if the set of evolutionary models for regular behaviors is not complete, or there exist anomalies that are not APT related.

A detection model that is capable to identify novel APT attacks is proposed in [29] by analyzing provenance graphs derived from system events. By using online metric learning (OML), their model learns a latent feature embedding by minimizing the distance between provenance subgraphs of the same class and maximizing the distance between

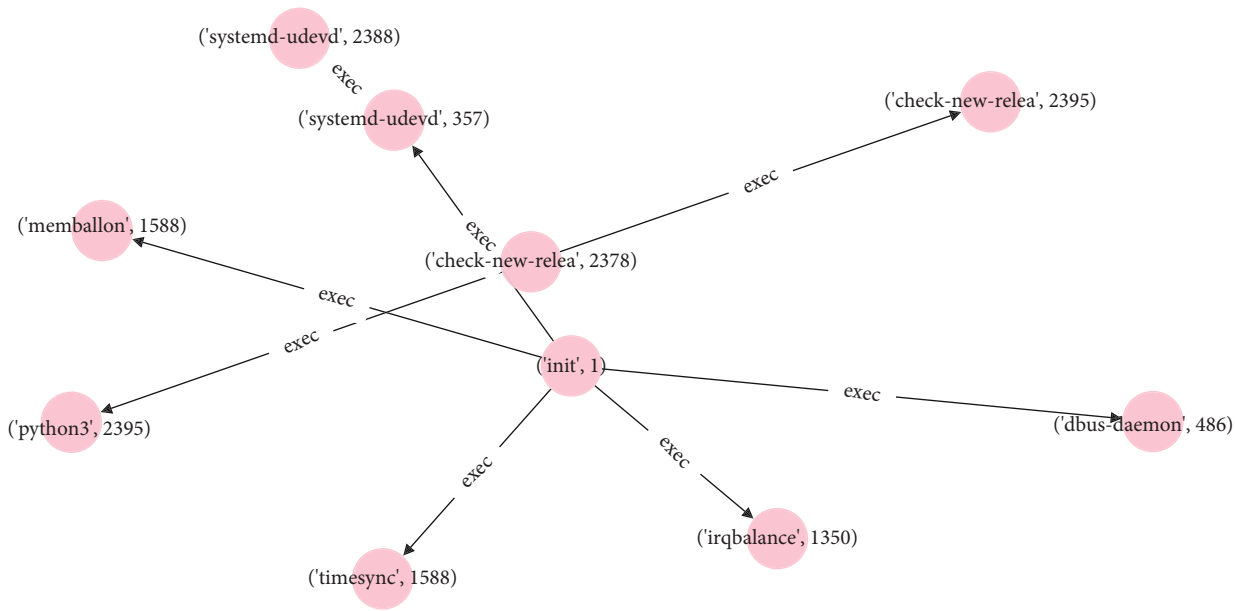


Figure 1: Provenance graph of a sample trace of system logs. Nodes represent system entities. Arrows and labels on edges indicate the direction and type of relations between entities.

subgraphs of different classes. However, using node2vec and the average over nodes to get the features of a graph in this paper only considers the neighboring information but omits the heterogeneity of nodes and edges. Additionally, the output of their model is either "benign" or "attacks," hence it cannot differentiate different stages of APT, though instances of different APT stages are labeled in their data.

There is no doubt that provenance graphs are an effective way to structure system logs and provide more informative data to improve the detection accuracy of APT attacks. However, more work needs to be performed to address some challenges in this area. First, appropriate graph-based learning approaches need to be applied to provenance graphs generated from system logs to efficiently detect APT by encoding important information such as the graph structure and properties of nodes and edges. Second, advanced APT can imitate benign activities or make use of benign applications to fulfill its goals, for example, by exploiting zero-day vulnerabilities of benign software. To detect such malicious behaviors of APT, lots of false positives will be incurred as APT alerts can also be triggered by benign activities. Therefore, strategic ways are needed to deal with false positives.

In this work, SR2APT, a novel host-based detection and response model against multistage APT attacks is proposed by analyzing system logs. To address the first challenge, an anomaly detection engine based on graph convolutional network (GCN) is built to identify different stages of APT and benign behaviors from subgraphs of a system log provenance graph. It improves detection accuracy by leveraging useful features hidden in the graph structure. To deal with the issue of false positives, a decision engine based on deep reinforcement learning is built, which provides optimal alert response actions such that actual APT attacks are prevented in earlier stages and minimal mistaken active defense actions are enforced on benign activities that generate false alerts.

3. Overview of Model Architecture

SR2APT can fit in the IDS architecture defined by the Intrusion Detection Working Group (IDWG), which consists of four modules: an event module, a database module, an analysis module, and a response module [30]. Figure 2 summarizes the functions of these modules and how functions are performed in SR2APT; the dashed arrows in it indicate important workflows including the inputs and the outputs of two engines that drive the APT detection and strategic response to alerts.

The event module incorporates a provenance graph generator, which dynamically processes system call events captured by monitoring tools such that the values of certain fields are properly extracted to update the causal relations on the provenance graph accordingly.

The database module stores all necessary information to support the functionalities of the analysis module, including

(i) A whole provenance graph that is updated dynamically; but the proposed APT detection engine does not take the entire graph as its input; instead, it analyzes focal subgraphs that are local structures of the entire graph. The subgraphs can be obtained by searching the neighborhood of a node of interest or a new node. In the experiments of this paper, a public dataset is used whose subgraphs are formed by querying specific node names backward a predefined number of steps. Each subgraph is sent to the detection engine in the analysis module. The detection result indicating the class of a subgraph is

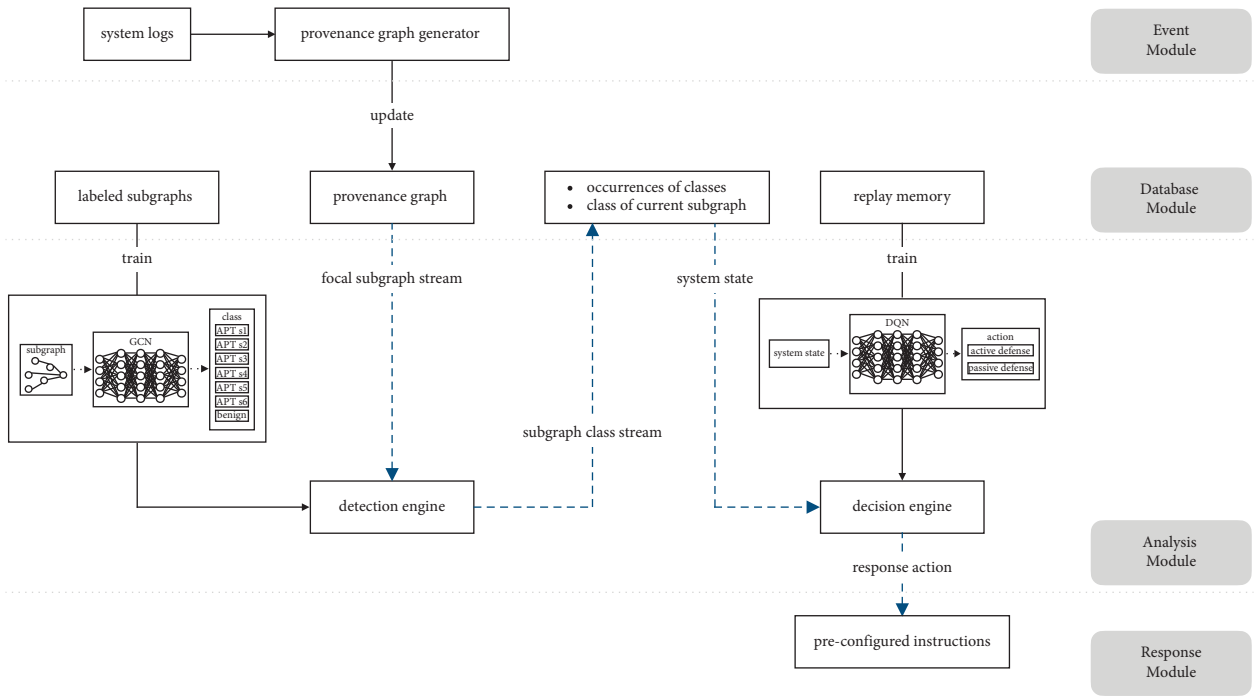


Figure 2: The functional modules and workflows in SR2APT. Blue dashed arrows indicate the input and output of the detection engine and decision engine.

stored in the database module, and the subgraph will be discarded after being evaluated.

(ii) Representative subgraphs with their class labels and abstracted representations including a node feature vector and an adjacency matrix; these subgraphs are used for training the graph convolutional network in the detection engine, and they can be periodically updated to adapt to new APT behavioral patterns.

(iii) Occurrences of all classes received from the detection engine within the current observing window, which will be used to form the state of the system together with the newest detection result. The reason for including historical counts of all classes is that they may exhibit different patterns when they are generated in attack scenarios and nonattack scenarios because APT stages usually appear more independently in nonattack scenarios than in attack scenarios [19].

(iv) Replay memory, which is a key term from the deep Q-network algorithm. It is used for training strategic response actions in the decision engine. Each entry in the replay memory is a 4-tuple: current system state, response action, next system state, and immediate reward.

The analysis module is where the intelligence lies, and it consists of a detection engine and a decision engine to effectively defend against multistage APT. The core of the detection engine is a graph convolutional network that classifies a graph as one of the multiple classes. In SR2APT, these classes include different stages of APT and a general benign class. The decision engine outputs the optimal response action to the APT alert based on the state of the system, which is formulated by deep reinforcement learning and is trained using the deep Q-network algorithm.

The response module automatically executes preconfigured instructions corresponding to the response action outputted from the decision engine in the analysis module. Since the main focuses of this work are novel modelings of the detection engine and the decision engine, a simplified set of response actions to APT alerts is considered in the experiments, in which only two types of responses are available: passive defense and active defense. Examples of preconfigured instructions in the response module for passive defense can be simply "no operation," and the instructions for active defense can be "stop process," "reset password," "reboot host," etc.

4. APT Detection and Response Model

In this section, technical details of the detection engine and decision engine in SR2APT are provided. For each part, the mathematical model used to formulate the problem is introduced first, followed by how it is integrated into SR2APT.

4.1. GCN-Based APT Stage Detection Engine

4.1.1. Graph Convolutional Network. To effectively classify a provenance graph generated from system logs, it is desired to encode more useful information such as the graph structure, properties of nodes, and edges. A graph convolutional network (GCN) model [31] is used in SR2APT to encode all this information to perform a multiclass classification task.

Additionally, it does not require prior knowledge to build the model, and it can handle graphs of various sizes directly.

The GCN model used in SR2APT is based on the eigen decomposition of the graph Laplacian matrix, and it addresses limitations of efficiency and the vertex localization problem in the first spectral-based graph convolutional network model [32], by truncating the Chebyshev polynomial to first order and relaxing the largest eigenvalue. More specifically, given an undirected graph $G = (V, E)$, where V is the set of nodes with $|V| = n$ and E is the set of edges with $|E| = m$, the adjacency matrix of the graph is denoted by A with $A_{ij} = w_{ij}$ where w_{ij} is the weight of the edge between node i and node j. If no edge exists between node i and node j, then $w_{ij} = 0$. For an undirected graph, A is symmetric. The degree diagonal matrix D is defined as $D_{ii} = \Sigma_{j=1}^{n} A_{ij}$, and the Laplacian matrix of the graph denoted by L is computed by $L = D - A$. Additionally, the normalized Laplacian matrix is computed by $\widetilde{L} = I - D^{-1/2}AD^{-1/2}$ where I is the identity matrix of size $n \times n$. In the GCN model, the input and output of each graph convolutional layer are a matrix X_l of size $n \times d_l$, which is the node feature representation where d_l is the dimension of the feature map defined for the layer l. The propagation operator applied to the graph convolutional layer in GCN is defined as follows:

$$X_l = \sigma\left(\overline{D}^{-1/2}\overline{AD}^{-1/2}X_{l-1}\Theta_l\right), \tag{1}$$

where X_l is the output node feature and X_{l-1} is the input node feature for the graph convolutional layer l. $\overline{A}$ is computed by $\overline{A} = A + I$ which adds self-loops to the adjacency matrix of the original graph. Accordingly, $\overline{D}$ is the degree diagonal matrix of $\overline{A}$, and Θ_l is the parameter matrix of size $d_{l-1} \times d_l$ which is learned in the model training phase. σ represents an activation function, which is usually a differentiable nonlinear function used in neural networks to learn more complex functions, e.g., ReLU, sigmoid, softmax, and tanh. X_0 is the node feature representation ingested by the first graph convolutional layer and the model input.

4.1.2. APT Stage Detection Engine. The GCN-based detection engine in SR2APT performs a multiclass classification task on provenance subgraphs. In the event module of SR2APT, system logs are structured in the form of a provenance graph using tools such as CamFlow [33], and GrAALF [34] is leveraged to build the subgraph dataset used for the experiments in this work. A valid system log record for a provenance graph should have two nodes with some identifiers and their relation. Taking GrAALF as an example, the fields "timestamp," "from_name," "from_id," "to_name," "to_id," and "event_type" are extracted from valid system log records. Then, based on the "id" information, the causal relations between nodes are established to form a provenance graph; "name" and "event_type" are labels of nodes and edges in the graph.

Subgraphs from a provenance graph are the neighboring structure around nodes of interest. Nodes of interest can be newly added nodes on a streaming provenance graph [21], specific processes, or files [34]. Then, the neighboring structure of a node can be obtained by using some search methods, for example, traversing backwards or forwards for a certain number of steps from the node. In the operation of SR2APT, the timestamp of nodes is used to make sure that subgraphs sent to the detection engine maintain the correct temporal order of system activities, which is important for solving optimal sequential response actions in the decision engine.

The labels of nodes and edges in a graph are encoded into numeric values, and a length threshold is applied to deal with long labels. The input of the GCN model, also known as the node feature X_0, is a vector of node labels. The values in the adjacency matrix of a graph, which is used for the operation of graph convolutional layers, are edge labels.

The GCN-based detection engine in the experiments of this paper consists of an input layer and three graph convolutional layers, and each of them is followed by a pooling layer, a dense layer, and an output layer. A cross-entropy loss function is used; therefore, the true label of a graph is represented as a one-hot vector. Seven classes are defined for the multiclass classification task, including six classes indicating APT stages 1, 2, 3, 4, 5, 6, and a general benign class. The model output is a 7-dimensional vector, which is a probability distribution over all seven classes, and the class with the largest probability is the predicted label of the input graph. The detection engine continuously transmits its outputs to the database module. If the current classification result is positive, indicating an alert of one APT stage, the current system state is then sent to the decision engine to decide the optimal response action to this alert.

4.2. DQN-Based APT Response Decision Engine

4.2.1. Deep Reinforcement Learning. The motivation of APT prevention in SR2APT is to strategically respond to APT alerts in a threat model where APT alerts can be triggered by both benign activities and APT attacks. Such a proactive defense approach can reduce the damage of APT by prohibiting it from evolving to deeper stages, which is more cost-effective than reactive approaches whose responses can be too late as APT might already or almost succeed at the moment of the defender's first response. However, enforcing active defense actions on the APT alerts triggered by benign activities incurs costs, which should be avoided as much as possible.

The decision engine in SR2APT considers APT characteristics (multistage, stealthiness, and advance), costs, and benefits of response actions to APT alerts from benign or APT attack activities. It then formulates the problem of strategic alert response based on deep reinforcement learning. The goal of a reinforcement learning agent is to learn a transformation from environment states to a probability distribution over feasible actions (also known as a policy) such that its cumulative reward is maximized [35]. By thoughtfully modeling the evolution of APT stages and the costs and benefits of response actions in different situations, a deep reinforcement learning model is able to drive the decision engine to generate an optimal response policy.

The mathematical formulation of reinforcement learning (RL) is based on a 5-tuple (S, A, R, P, γ); S is a set of environmental states that can be observed by the agent. A is a set of actions that the agent can take in each state. R is a set of rewards received by the agent after taking an action. P is a state transition model represented by a matrix where P_{ij} is the probability of transitioning from state i to state j. γ is the discount factor that controls the importance of future rewards in the characterization of the agent's cumulative reward. The goal of RL is to learn the optimal policy in each state. Since the system state defined in the decision engine of SR2APT is high-dimensional, a deep reinforcement learning algorithm is used, in which a neural network learns a function to map system states to a distribution over response actions.

In SR2APT, the defender is the only RL agent, but it can be extended to a multiagent problem by considering the attacker as another strategic agent. Figure 3 illustrates the abstracted interactions between an RL agent and the environment in a deep reinforcement learning setting. At time t, the agent takes action a_t in state s_t, then the environment transitions to the next state s_{t+1} and the agent receives its reward r_t of taking that action. As this is a sequential decision-making process and the action taken in the current state not only affects the immediate reward but also affects future rewards through state transition, the objective of the RL agent is to maximize the cumulative reward (also called value function) starting from the current state that is formulated as follows:

$$V(s_t) = r(s_t) + \gamma r(s_{t+1}) + \gamma^2 r(s_{t+2}) + \ldots + \gamma^{T-t} r(s_T). \quad (2)$$

If the action of the agent is taken as another independent variable, a state-action value function (also called the Q function) is formulated as follows, where $P(s'_{t+1}|s_t, a_t)$ is the probability of transitioning to state s'_{t+1}, when the agent takes action a_t in the state s_t:

$$Q(s_t, a_t) = r(s_t, a_t) + \gamma \sum_{s'_{t+1}} P(s'_{t+1}|s_t, a_t) \cdot V(s'_{t+1}). \quad (3)$$

Therefore, the objective function of the agent can be reformulated as follows:

$$\max_{\pi(s)} \sum_a Q(s, a) \cdot \pi(a), \quad (4)$$

where $\pi(s)$ is called a policy, which is a probability distribution over all feasible actions of the agent in the state s.

The deep Q-network (DQN) algorithm [36] is used to train the decision engine. DQN uses a deep convolution neural network called Q-network to approximate the state-action value function $Q(s, a, \theta)$. The advantages of this technique are twofold. First, it leverages a replay mechanism by randomly sampling a mini batch from a replay memory to train the Q-network for every update. Each data unit stored in the replay memory is represented by a 4-tuple (s_t, a_t, r_t, s_{t+1}), meaning that taking action a_t in state s_t leads to an immediate reward r_t and state s_{t+1}, and it can be viewed as an "experience." Second, a slowly updated neural network called a target network $Q(s, a, \theta')$ is used to compute the target value Q-network in every iteration. The parameters of the target network are copied from the Q-network periodically, and they are fixed in each iteration. These two special designs ensure the stability of the DQN algorithm, smooth out the learning process, and effectively avoid divergence.

The DQN algorithm is implemented as follows. Before training, a replay memory of fixed-length, which stores the most recent experiences, a Q-network $Q(s, a, \theta)$, and a target network $Q(s, a, \theta')$ with the same random parameters $Q(s, a, \theta')$ are initialized. In the beginning of every iteration i, the agent is in a state s_t, then it chooses an action a_t based on the rule of $\varepsilon -$ greedy which balances exploitation (stick to the current best action) and exploration (randomly try other actions). After taking the action, the agent receives reward r_t and learns that the state transitions to s_{t+1}. The 4-tuple (s_t, a_t, r_t, s_{t+1}) is then stored in the replay memory. Once the size of the replay memory is greater than the predefined batch size, a batch of 4-tuples is sampled randomly from the replay memory to update the parameters of the Q-network according to the loss function as follows:

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim B}\left[\left(r + \max_{a'} Q\left(s', a', \theta'\right) - Q(s, a, \theta_i)\right)^2\right], \quad (5)$$

where B is the sampled batch, the first $Q(\cdot)$ is the target network, and the second $Q(\cdot)$ is the Q-network. Note that, if s' is a terminal state, the term $\max_{a'} Q(s', a', \theta')$ in the loss function will not exist; therefore, the Q-network needs to be trained by a new episode of interaction.

4.2.2. Decision Engine. The problem setting in the decision engine of SR2APT is as follows. APT attackers intend to hide their behaviors by imitating benign activities, for example, the data exfiltration stage in APT might look like a normal file transfer behavior by an authenticated user. Therefore, the threat model considered in this paper is that APT-related alerts from the detection engine can be triggered by APT attacks or benign activities. For the sake of brevity, in the rest of the paper, actual APT activities were referred to as AAPT, and the complementary benign activities imitated by APT were referred to as BAPT. By constructing dedicated cost/reward functions based on the understanding of APT attacks, a strategic alert response policy can be generated by the decision engine such that AAPT is effectively impeded in earlier stages while less BAPT is mistakenly impeded, without explicitly knowing if an APT alert is triggered by AAPT or BAPT. To integrate APT characteristics into the decision engine, the effectiveness of active defense in stopping APT movement is assumed to be limited, as APT attackers are capable to exploit zero-day vulnerabilities. The experiments that evaluate the performance of the decision engine in this paper focus on a six-stage APT attack; however, it can be fitted into other APT attack scenarios by modifying the parameters in the deep reinforcement learning model. States, actions, rewards, and state transitions defined in the decision engine are described in detail as follows.

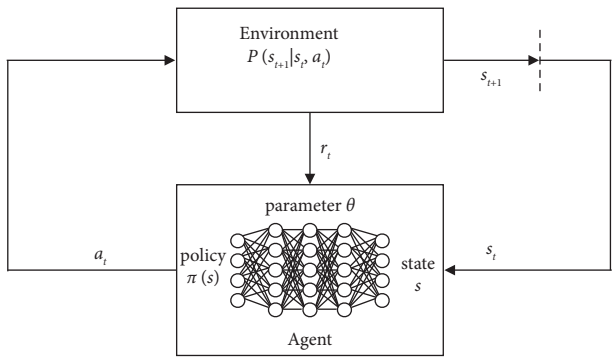


Figure 3: Interactions between agent and environment in deep reinforcement learning (a: action; r: reward; s: state; P: state transition model).

(1) States. The system states are constructed by fusing the classification results from the detection engine from the beginning of the current monitoring episode. It is represented by an 8-dimensional vector $s = (c_0, c_1, c_2, c_3, c_4, c_5, c_6, cs)$ where c_0 is the number of occurrences of the benign class, $c_i\,(i \in [1,6])$ is the number of occurrences of each APT s_i class, and cs indicates the current signal (classification result). Each episode starts from a system state where an alert of APT $s_i\,(i \in [1,5])$ is raised, for example, an initial system state can be $s = (0,0,1,0,0,0,0,2)$ which means the current signal is APT s_2 class and the occurrence of APT s_2 class is 1 while the occurrence of other classes is 0.

In the threat model considered by this paper, either the APT attacker or the defender takes control of the host in a monitoring episode. Therefore, terminal states are different depending on who is taking over the host. When the attacker is in control, a sequence of detection results that the decision engine receives is regarded as an AAPT alert trace. Terminal states are those states in which the current signal is 6 (indicating the attacker wins to the end), or the current signal is 0 and it results from the defender's active defense (indicating the defender takes over again), or the maximum length of a monitoring episode is reached. When the defender is in control, a sequence of detection results is regarded as a BAPT alert trace, and the only terminal state is the state in which the maximum length of a monitoring episode is reached.

(2) Actions. For nonterminal states, a defender can choose from two response actions: passive defense and active defense. In practice, examples of active defense are to investigate suspicious behaviors, patch vulnerabilities, reboot machines, and reset passwords; passive defense can be simply ignoring the alert, either because of not trusting it or for the purpose of continuous observation.

(3) Rewards. After taking a response action, the decision engine will receive a reward as feedback from the environment. In SR2APT, this reward is dependent on the current detection result and whether it is part of an alert trace of AAPT or BAPT and whether the action is active defense or passive defense. Since earlier APT stages usually have limited damage than later stages, an exponential function is used to quantify the damage of an APT stage and the cost of active defense to impede its movement. For example, if activities of an AAPT are being monitored and current detection result is APT stage s_i, the damage of the behavior that triggers this alert is $d^{s_i}\,(d > 0)$ and the cost of active defense against it is $c^{s_i}\,(c > 0)$, then the reward of choosing active defense action is $d^{s_i} - c^{s_i}\,(d > c)$ while the reward of choosing passive defense action is $-d^{s_i}$, However, if active defense action is enforced on the APT alerts triggered by benign activities, the defender receives a penalty. In SR2APT, this penalty is also an exponential function of APT stage, that is, if current detection result is APT stage s_i which is triggered by a BAPT, then the reward of taking active defense action is $-p^{s_i}\,(p > 0)$ while the reward is 0 by taking passive defense action.

(4) State Transition. Deep reinforcement learning allows to model real-world uncertainties via probabilistic state transitions. In SR2APT, state transitions are decided by the ordering of detection results, which is affected by the interdependencies between APT stages conditioning on the defender's responses to APT alerts. In this paper, the effectiveness of active defense action against APT is assumed to be limited, in order to model an APT attacker's ability to exploit zero-day vulnerabilities. In addition, the ordering of different detection results in an AAPT alert trace and a BAPT alert trace should exhibit different patterns because APT stages are more interdependent in AAPT but not in BAPT. It would be ideal to use real AAPT alert traces and BAPT alert traces to train the decision engine, but synthetic alert traces also work if real data is not available as long as the synthetic traces can represent the behavioral patterns of AAPT and BAPT. In the experiments of this paper, synthetic alert traces are used and they are generated based on the following behavioral patterns of AAPT and BAPT.

(i) Current detection result is the benign class. No matter if it is part of AAPT or BAPT, the defender does not respond to it. Therefore, if it is part of AAPT, then it indicates that the attacker is staying quiet, and the next expected detection result is the benign class or a higher APT stage according to the attacker's movement plan. If it is part of BAPT, the next expected detection result is one of the seven classes defined in the detection engine at random.

(ii) Current detection result is an APT stage s_i and it is part of AAPT. When the defender responds by taking a passive defense action, the attacker will stay quiet for a random period of time and then evolve to stage s_{i+1}, When the defender takes an active defense action, the movement of APT would be effectively impeded with probability $e\,(0 < e < 1)$, in this case, the attacker will either be kicked out of the defender's system or stay quiet for a while then restart from an earlier stage $s_j\,(j \in [1, i))$. However, the active defense could also be ineffective with probability $(1 - e)$, in this case, the attacker will stay quiet for a while then evolve to stage s_{i+1}.

(iii) Current detection result is an APT stage s_i and it is part of BAPT. No matter what response action the defender takes, the next expected detection result is either the benign class or APT stage s_j ($j \in [1, 6]$) at random.

5. Evaluation

In this section, the proof of concept experiments are designed and implemented to evaluate the performance of SR2APT in defending against APT. First, a semisynthetic dataset of system log provenance graphs is used to evaluate the performance of the GCN-based detection engine in the subgraph classification. Then, synthetic alert traces from both APT and benign scenarios are used to compare the performance of the strategic alert response policy from the DQN-based decision engine and two baseline fixed response policies.

5.1. Provenance Subgraph Classification. The dataset of system log provenance graphs from [18] is used to train and test the detection engine, which covers the system logs of system activities about six APT stages: system reconnaissance, network discovery, persistence, privilege escalation, asset discovery, and data exfiltration; as well as some benign system activities.

To construct the dataset, first Sysdig is used to monitor the activities on a host and generate system logs, which allows up to 57 fields to structure raw log data. Then, GrAALF [34] is used to get subgraph samples for each class in the detection engine. It is a Java application that can process raw system logs, store processed data in a graph database, and support queries to retrieve subgraphs. Regarding the dataset used in the experiment of this paper, nodes of interest are predefined for each APT stage behavior e.g., "whoami" is a keyword for the reconnaissance stage [18]. Then a subgraph sample for an APT stage is obtained by starting from the nodes of interest to traverse backwards or forwards for certain steps. For instance, Figure 4 shows the subgraph data returned by GrAALF, which corresponds to the APT privilege escalation stage, by running the query "*back select * from * where name is su*." Eventually, 300 subgraphs for each of the seven classes are obtained by using the provided query templates and resampling. The seven classes in the experiments are benign (class 0), APT s1 (class 1), APT s2 (class 2), APT s3 (class 3), APT s4 (class 4), APT s5 (class 5), and APT s6 (class 6). Out of total 2100 subgraphs, 80% are used for training the detection engine, 10% are used for validation, and 10% are used for testing.

Figure 5 shows the confusion matrix generated by comparing the true labels and predicted labels of 210 tested subgraphs. The X-axis and Y-axis are labeled by the seven classes defined in the detection engine, and the value at row i and column j is the number of subgraphs whose true label is class i and predicted label is class j. As shown in Figure 5, the values at off-diagonal positions are much smaller than the values at diagonal positions for every row and column, meaning that the detection engine of SR2APT can effectively identify malicious behaviors of different APT stages and benign behaviors, with overall 94% classification accuracy, which outperforms other compared classification models SVM (83%), CNN (84%) and LSTM (86%).

The basic metrics for evaluating the performance of a classifier are true positive (TP), true negative (TN), false positive (FP), and false negative (FN). TP is the number of positive instances that are correctly classified as positive. TN is the number of negative instances that are correctly classified as negative. FP is the number of negative instances that are incorrectly classified as positive. FN is the number of positive instances that are incorrectly classified as negative. Other important metrics including the true positive rate (TPR), false positive rate (FPR), precision, and recall are calculated as follows:

$$\begin{aligned} \mathrm{TPR} &= \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}, \\ \mathrm{FPR} &= \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}, \\ \mathrm{precision} &= \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}, \\ \mathrm{recall} &= \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}. \end{aligned} \tag{6}$$

The AUC-ROC curve of each class classified by the detection engine of SR2APT is shown in Figure 6, which evaluates the performance of the proposed detection engine in classifying each class, under different probability threshold settings. When the probability threshold changes, it could be the case that both the TPR and the FPR change or only one of them changes, thus producing vertical, horizontal, and slant segments on an AUC-ROC curve. The black straight dashed line is a benchmark indicating a model has no classification power. Figure 6 shows the AUC (area under the curve) of each class classified by the detection engine is almost 1.00, meaning that the detection engine is not sensitive to the probability threshold, and the learned feature representation for each class is effective.

$$\mathrm{F1-score} = \frac{2 \cdot \mathrm{Precision} \cdot \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}} = \frac{\mathrm{TP}}{\mathrm{TP} + 1/2\,(\mathrm{FN} + \mathrm{FP})}. \tag{7}$$

Additionally, the F1-score is used to compare the proposed GCN-based detection engine with some classic classification models, because it evaluates a model by considering both FP and FN. From Equation (7), the F1-score is the harmonic mean of precision and recall, where precision reflects the impact of FP and recall reflects the impact of FN. The F1-score of a model is high if both precision and recall are high, and is low if one or both of them are low. As the detection results are important signals to the decision engine, both FP and FN are not desired, thus the F1-score is a good metric to compare detection models using different kernels. Three commonly used kernels of classification models for anomaly detection are tested on the same dataset, which are support vector machine (SVM), one-dimensional

```
{ "sequence_number": 42578,
  "user": "boba_fett",
  "from_id": 2402 ,
  "from_name": "perl",
  "evt_type": "exec",
  "to_name": "sh" ,
  "to_id": 2665 ,
  "count": 1},

{ "sequence_number": 42592,
  "user": "boba_fett",
  "from_id": 2665 ,
  "from_name": "sh",
  "evt_type": "exec",
  "to_name": "fl7Klh5a" ,
  "to_id": 2666 ,
  "count": 1},

{ "sequence_number": 43311,
  "user": "boba_fett",
  "from_id": 2666 ,
  "from_name": "fl7Klh5a",
  "evt_type": "exec",
  "to_name": "su" ,
  "to_id": 2679 ,
  "count": 1}
```


Figure 4: A sequence of system logs for an APT privilege escalation stage.

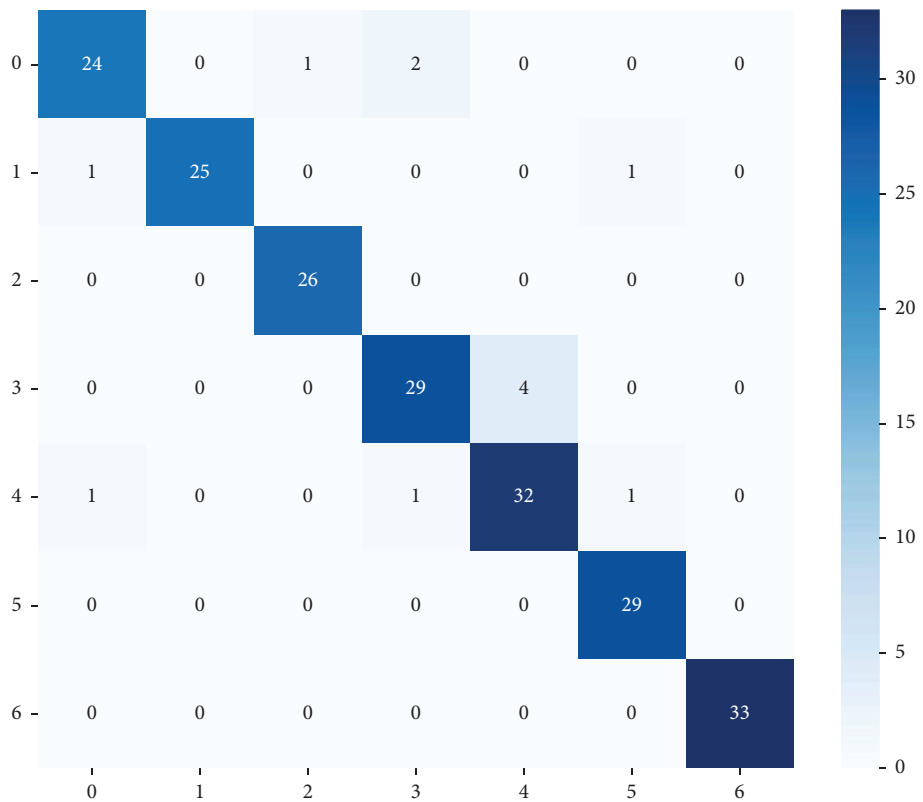


Figure 5: The confusion matrix for the tested provenance subgraphs.

convolutional neural network (CNN), and long short-term memory (LSTM). To implement the kernels other than GCN, "from_name," "evt_type," and "to_name" in the sequence of system logs corresponding to a subgraph are extracted in temporal order to form the input of classification models. The parameters of every model are tuned such that it achieves the highest accuracy. The sklearn Python library is used to implement the linear SVM model. The Keras Python library is used to implement the one-dimensional CNN model and the LSTM model. The one-dimensional CNN consists of one 1D convolutional layer (filters = 20, kernel size = 3), one 1D max pooling layer (pool size = 3), and one dense layer with softmax activation function being used. For the LSTM neural network, it consists of one masking layer, three LSTM layers with each followed by a dropout layer (dropout rate = 0.2), and one dense layer using the softmax activation function. To train the one-dimensional CNN model and the LSTM model, the Adam optimizer and categorical cross-entropy loss function are used. As shown in Table 1, the GCN-based detection

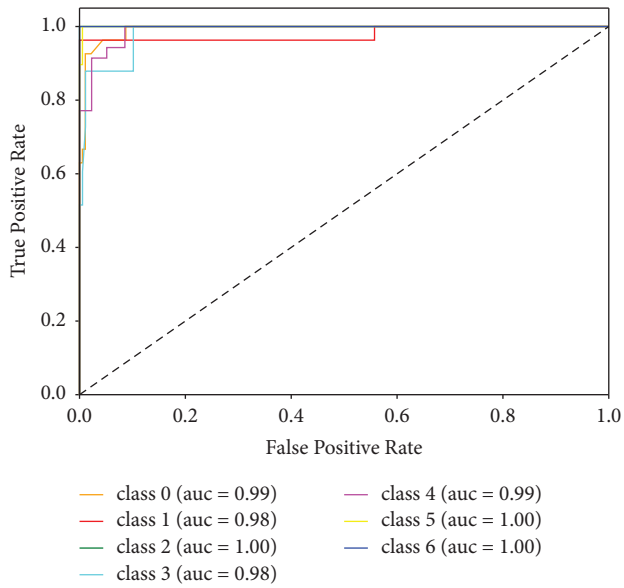


Figure 6: The AUC-ROC curve for each class classified by the detection engine of SR2APT.

Table 1: The F1-score of classification models based on SVM, CNN, LSTM, and GCN regarding each class.

	Benign	APT s1	APT s2	APT s3	APT s4	APT s5	APT s6
SVM	1.00	0.52	0.96	0.80	0.73	0.89	0.89
CNN	**0.98**	0.65	0.95	0.72	0.77	0.89	0.90
LSTM	0.95	0.62	**1.00**	0.88	0.87	0.78	0.87
GCN	0.91	**0.96**	0.98	**0.89**	**0.90**	**0.97**	**1.00**

engine achieves the highest F1-score for classes of APT s1, APT s3, APT s4, APT s5, and APT s6. For the other two classes of benign and APT s2, the GCN-based detection engine still achieves an F1-score that is greater than 0.9. Therefore, GCN improves the performance of the detection engine in provenance subgraph classification compared to SVM, CNN, and LSTM, because it leverages useful features hidden in the dependencies of nodes in a graph.

5.2. Strategic Response Policy to APT Alerts. To train and evaluate the performance of the decision engine that outputs strategic response actions to sequential APT alerts within a monitoring window, synthetic alert traces for APT scenarios (AAPT) and benign scenarios (BAPT) are used. These synthetic alert traces are generated according to the behavioral patterns of the APT attack considered in this paper, which are affected by the defender's response actions. More details about how synthetic alert traces are generated are provided in the state transition part of Section 4.

When implementing the proposed decision engine, preliminary works are needed to provide gain/loss feedback after taking a response action, which is host-dependent and is out of the scope of this paper. To generate the feedback information for the proof of concept experiments, the parameter settings used in the experiments are: $d = 4$, $c = 1.5$, $p = 1.5$ which are the base in the exponential function representing the damage of each APT stage, the cost of active defenses against each APT stage, and penalty of mistaken active defenses to APT alerts triggered by benign activities, respectively; $e = 0.9$ which indicates the effectiveness rate of active defenses in slowing down the movement of APT. Note that the parameters in the experiments are chosen only because they allow the total reward of the DQN response policy to be positive, but they can be other values. The advantage of the DQN policy is not dependent on parameter settings, rather it is because the DQN policy is learned from trial and error. Therefore, with sufficient experiences the DQN policy is always able to approximate the optimal policy. When implementing the decision engine, the defender should use their own feedback mechanism based on the status of the protected target, rather than the parameter settings used in the proof of concept experiments.

To demonstrate the advantages of the response policy generated by the decision engine (DQN policy) in the SR2APT model, the DQN policy is compared with two fixed rate response policies. A fixed rate policy p is defined as follows: every time an APT alert is reported, a defender takes an active defense action with a fixed probability p. One baseline fixed rate policy is $p = 1.0$ which represents the most aggressive response policy, and the other fixed rate policy is $p = 0.5$.

The decision engine was trained for 100 episodes. Each episode simulates an alert trace of AAPT or BAPT, which is dynamically affected by the defender's response actions. Before the start of an episode, whether this episode simulates AAPT or BAPT is decided by the ratio of AAPT: BAPT = 3 : 7, which represents the defender's perception of how often

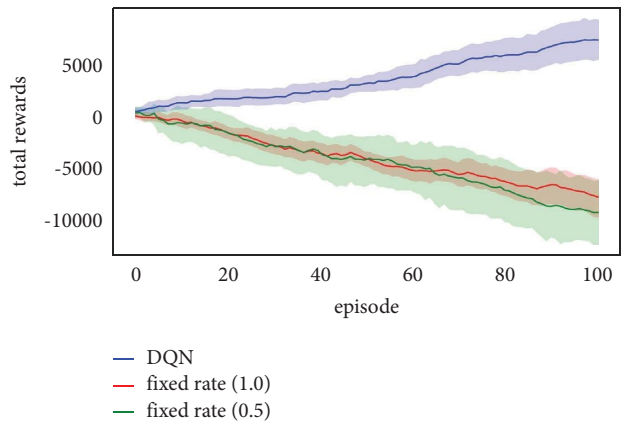


Figure 7: Total rewards by following the DQN response policy from the decision engine of SR2APT and two fixed response policies.

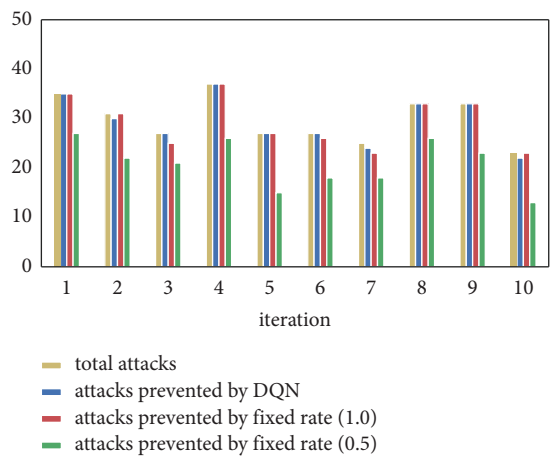


Figure 8: Number of APT attacks prevented by following the DQN response policy from the decision engine of SR2APT and two fixed response policies.

the system is taken over by the attacker and the defender. Once the decision engine is trained, the DQN response policy is evaluated by running the decision engine for 10 iterations, where each iteration consists of 100 episodes.

First, the total rewards are compared by following the DQN alert response policy from the decision engine and two fixed rate policies. As shown in Figure 7, the two fixed rate policies perform equally worse than the DQN policy. Although a more aggressive fixed rate policy can prevent damage from APT activities by frequently taking active defenses, it also increases the probability of mistaken active defenses to false APT alerts triggered by benign activities, and vice versa. However, the DQN policy learns to implicitly treat the APT alerts triggered by AAPT and BAPT differently, to minimize the penalties from mistakenly impeding BAPT and maximize the gains from slowing down the movement of AAPT by taking active defenses.

Next, the ability of each response policy to successfully prevent AAPT is evaluated. Here, a successful prevention against AAPT means that the APT attacker is defeated out of the defender's system. In Figure 8, the yellow bar represents the total number of APT attacks in each iteration, and the other three bars represent the number of attacks that are successfully prevented by following the three response policies, respectively. As shown in Figure 8, the less aggressive fixed rate policy ($p = 0.5$) is the worst, but both the DQN policy and the most aggressive fixed rate policy ($p = 1.0$) are able to prevent almost all attacks in every iteration. Since the most aggressive fixed rate policy instructs the defender to take an active defense action against all APT alerts, it is difficult for an APT attack to succeed under this circumstance. Although the DQN response policy from the decision engine treats APT alerts differently, it still performs extremely well to prevent attacks, and it achieves greater total rewards than the most aggressive fixed rate policy.

To further address the advantage of the strategic DQN response policy, the number of mistaken active defenses enforced on BAPT by following the three policies are collected. In Figure 9, each colored bar represents the total number of active defense actions enforced on the APT alerts triggered by BAPT by following the corresponding policy. It shows that the most aggressive fixed rate policy ($p = 1.0$) is the worst, as it instructs the defender to take active defense actions for all APT alerts from BAPT, which is not a wise policy for the considered threat model, where BAPT alert traces appear more frequently than AAPT. The less

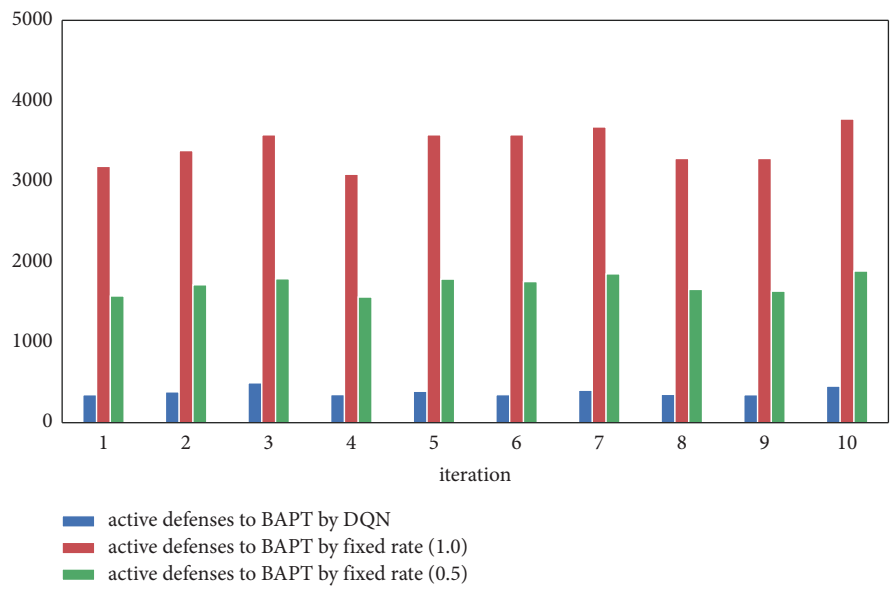


Figure 9: Number of active defense actions enforced on benign activities by following the DQN response policy from the decision engine of SR2APT and two fixed response policies.

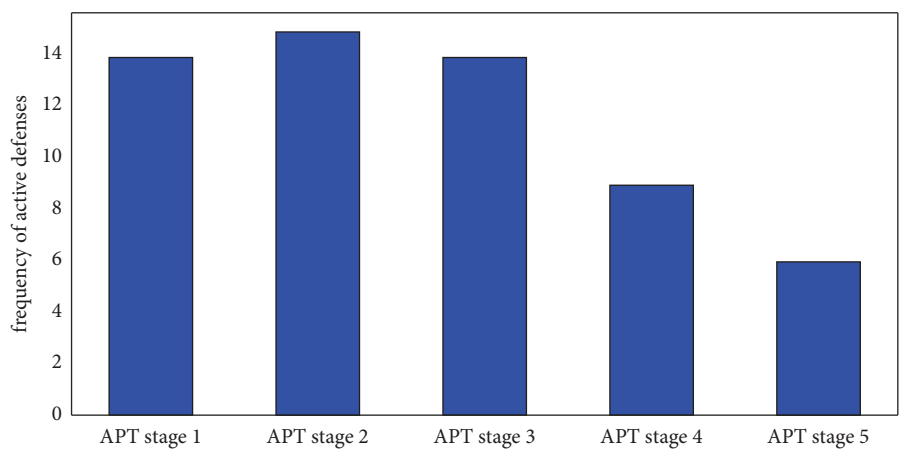


Figure 10: Average frequencies of active defense actions to APT stages of 1 to 5 in APT scenarios.

aggressive fixed rate policy ($p = 0.5$) reduces the number of mistaken active defenses to benign activities, but it is still much worse than the DQN policy. As the system states in the decision engine take historical detection results into consideration, it is likely that the system states for an alert from AAPT and an alert from BAPT are quite different because of the interdependencies between APT stages; hence, the DQN policy will choose not to take an active defense action to an alert triggered by BAPT as much as possible.

Finally, the strategic DQN alert response policy from the decision engine prevents APT attacks in earlier stages. The average frequencies of active defense actions in one iteration are collected for each APT stage, except for the APT stage s_6, which is an indicator of a terminal state where no response action is available. As shown in Figure 10, more active defense actions are enforced on the alerts corresponding to earlier stages of APT attacks, which is as expected because greater damages and costs of defense are assigned to later APT stages in the considered threat model.

Overall, the strategic APT alert response policy generated by the decision engine in the SR2APT model achieves the best performance compared with two fixed rate response policies, regarding the total rewards in the defense against APT, the ability to successfully prevent APT attacks, and mistaken active defenses for false APT alerts triggered by benign activities.

6. Conclusion

In this paper, a novel host-based APT detection and response model called SR2APT is proposed, which can be integrated into cyber-physical systems to provide cost-effective and agile protection to critical hosts against sophisticated APT attacks. The proposed model addresses two important but overlooked problems in the defense against multistage APT: preventing APT attacks in earlier stages and minimizing the impediments of mistaken response actions to benign

activities that generate false alerts. SR2APT tackles these two problems through a strategic APT alert response policy.

The detection engine of SR2APT applies a graph convolutional network to analyze subgraphs of a provenance graph generated from system logs, to detect malicious behaviors of each APT stage as well as benign behaviors. The experimental results on a semisynthetic dataset show that the GCN-based detection engine obtains 94% classification accuracy and outperforms classification models based on other kernels including SVM, CNN, and LSTM. To derive optimal response actions to sequential APT alerts from the detection engine, the decision engine of SR2APT is built upon deep reinforcement learning, which considers the characteristics of APT attacks such as multistage, stealthiness, and zero-day exploitation capability. With dedicated cost and reward functions in the decision engine such that they are compatible with the threat model considered in this paper, the alert response policy generated by the decision engine outperforms two baseline fixed response policies and achieves the highest total rewards in the defense against APT. Because it optimally balances the trade-off between maximizing the number of prevented APT attacks and minimizing the impediments to benign activities from mistaken active defense actions. In addition, the experimental results show that the strategic alert response policy from the decision engine is able to prevent APT attacks in earlier stages by taking advantage of deep reinforcement learning.

SR2APT can be applied to other APT scenarios, because both its detection engine and decision engine are flexible. This work focuses on multistage APT attacks on a host, thus a potential extension is to apply SR2APT to defend against APT attacks moving across multiple hosts. In that case, host-level system logs need to be replaced by network-level monitoring data, e.g., network traffic in the detection engine, to detect the lateral movement stage of APT. In addition, strategic alert responses in different situations where the defender has incomplete monitoring information or limited defense resources are also promising directions of the future work.

Data Availability

The host-level system logs data and synthetic APT alert traces for both APT and benign scenarios used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

Publication of this article was funded by the University of Colorado Boulder Libraries Open Access Fund.

References

[1] S. Han, M. Xie, H. Chen, and Y. Ling, "Intrusion detection in cyber-physical systems: techniques and challenges," *IEEE Systems Journal*, vol. 8, no. 4, pp. 1052–1062, 2014.

[2] L. Huang and Q. Zhu, "A dynamic games approach to proactive defense strategies against Advanced Persistent Threats in cyber-physical systems," *Computers & Security*, vol. 89, Article ID 101660, 2020.

[3] R. Langner, "Stuxnet: dissecting a cyberwarfare weapon," *IEEE Security and Privacy Magazine*, vol. 9, no. 3, pp. 49–51, 2011.

[4] A. Alshamrani, S. Myneni, A. Chowdhary, and D. Huang, "A survey on advanced persistent threats: techniques, solutions, challenges, and research opportunities," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1851–1877, 2019.

[5] A. Juels and T. Yen, "Sherlock Holmes and the case of the advanced persistent threat," in *Proceedings of the 5th USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET 12)*, San Jose, CA, USA, Apr. 2012.

[6] A. Rot and B. Olszewski, "Advanced persistent threats attacks in cyberspace. Threats, vulnerabilities, methods of protection," in *Proc. FedCSIS (Position Papers)*, pp. 113–117, Czech Republic, Prague, 2017.

[7] D. Gonzales, J. M. Kaplan, E. Saltzman, Z. Winkelman, and D. Woods, "Cloud-Trust: a security assessment model for infrastructure as a service (IaaS) clouds," *IEEE Transactions on Cloud Computing*, vol. 5, no. 3, pp. 523–536, 2017.

[8] R. Mitchell and I. R. Chen, "A survey of intrusion detection techniques for cyber-physical systems," *ACM Computing Surveys*, vol. 46, no. 4, pp. 1–29, Apr. 2014.

[9] P. N. Bahrami, "Cyber kill chain-based taxonomy of advanced persistent threat actors: analogy of tactics, techniques, and procedures," *Journal of Information Processing Systems*, vol. 15, no. 4, pp. 865–889, Aug. 2019.

[10] T. M. Chen and S. Abu-Nimeh, "Lessons from Stuxnet," *Computer*, vol. 44, no. 4, pp. 91–93, 2011.

[11] N. Villeneuve and J. Bennett, "Detecting APT activity with network traffic analysis," *Trend Micro Inc*, vol. 547, p. 78, 2012 Available at: http://www.trendmicro.com/cloud-content/us/pdfs/securityintelligence/.

[12] X. Wang, "Detection of command and control in advanced persistent threat based on independent access," in *Proc. IEEE Int. Conf. Commun. (ICC)*, pp. 1–6, Kuala Lumpur, Malaysia, May 2016.

[13] M. Marchetti, F. Pierazzi, M. Colajanni, and A. Guido, "Analysis of high volumes of network traffic for advanced persistent threat detection," *Computer Networks*, vol. 109, pp. 127–141, Nov. 2016.

[14] N. Nissim, A. Cohen, C. Glezer, and Y. Elovici, "Detection of malicious PDF files and directions for enhancements: a state-of-the art survey," *Computers & Security*, vol. 48, pp. 246–266, Feb. 2015.

[15] J. V. Chandra, N. Challa, and S. K. Pasupuleti, "A practical approach to E-mail spam filters to protect data from advanced persistent threat," in *Proc. 2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT)*, pp. 1–5, Nagercoil, India, Mar2016.

[16] S. Chandran, P. Hrudya, and P. Poornachandran, "An efficient classification model for detecting advanced persistent threat," in *Proc. 2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 2001–2009, Kochi, India, Aug. 2015.

[17] N. Mohamed and B. Belaton, "SBI model for the detection of advanced persistent threat based on strange behavior of using credential dumping technique," *IEEE Access*, vol. 9, pp. 42919–42932, 2021.

[18] M. AbuOdeh, "A novel AI-based methodology for identifying cyber attacks in honey pots," in *Proc. AAAI Conference on Artificial Intelligence*, pp. 15224–15231, Feb. 2021.

[19] J. Sexton, C. Storlie, and J. Neil, "Attack chain detection," *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 8, no. 5-6, pp. 353–363, Aug. 2015.

[20] P. Giura and W. Wang, "A context-based detection framework for advanced persistent threats," in *Proceedings of the 2012 International Conference on Cyber Security*, pp. 69–74, Alexandria, VA, USA, Dec. 2012.

[21] M. M. Sadegh, "Holmes: real-time apt detection through correlation of suspicious information flows," in *Proceedings of the 2019 IEEE Symposium on Security and Privacy*, pp. 1137–1152, San Francisco, CA, USA, May 2019.

[22] H. Studiawan, F. Sohel, and C. Payne, "A survey on forensic investigation of operating system logs," *Digital Investigation*, vol. 29, pp. 1–20, Jun. 2019.

[23] A. Awad, "Data leakage detection using system call provenance," in *Proceedings of the 2016 International Conference on Intelligent Networking and Collaborative Systems (INCoS)*, pp. 486–491, Ostrava, Czech Republic, September 2016.

[24] S. A. Hofmeyr, S. Forrest, and A. Somayaji, "Intrusion detection using sequences of system calls," *Journal of Computer Security*, vol. 6, no. 3, pp. 151–180, 1998.

[25] Y. Liao and V. R. Vemuri, "Using text categorization techniques for intrusion detection," in *Proceedings of the 11th USENIX Security Symposium*, San Francisco, CA, USA, Aug. 2002.

[26] B. Subba, S. Biswas, and S. Karmakar, "Host based intrusion detection system using frequency analysis of n-gram terms," in *Proceedings of the 2017 IEEE Region Ten Conference (TENCON)*, pp. 2006–2011, Penang, Malaysia, Nov. 2017.

[27] M. Anjum, S. Iqbal, and B. Hamelin, "ANUBIS: a provenance graph-based framework for advanced persistent threat detection," Dec. 2021, Available at: https://arxiv.org/abs/2112.11032.

[28] X. Han, "Unicorn: runtime provenance-based detector for advanced persistent threats," in *Proceedings of the 27th Annual Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, USA, Feb. 2020.

[29] G. Ayoade, "Evolving advanced persistent threat detection using provenance graph and metric learning," in *Proceedings of the 2020 IEEE Conference on Communications and Network Security (CNS)*, pp. 1–9, Avignon, France, Feb 2020.

[30] V. Jyothsna, V. Rama Prasad, and K. Munivara Prasad, "A review of anomaly based intrusion detection systems," *International Journal of Computer Application*, vol. 28, no. 7, pp. 26–35, Aug. 2011.

[31] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proceedings of the 2017 International Conference on Learning Representations*, pp. 1–14, Toulon, France, Apr. 2017.

[32] J. Bruna, "Spectral networks and locally connected networks on graphs," in *Proceedings of the 2nd International Conference on Learning Representations*, Banff, Canada, Apr. 2014.

[33] T. Pasquier, "Practical whole-system provenance capture," in *Proceedings of the 2017 Symposium on Cloud Computing*, pp. 405–418, 2017.

[34] O. Setayeshfar, C. Adkins, M. Jones, K. H. Lee, and P. Doshi, "Graalf: supporting graphical analysis of audit logs for forensics," *Software Impacts*, vol. 8, Article ID 100068, 2021.

[35] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT press, Cambridge, MA, USA, 2018.

[36] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

Hindawi
Security and Communication Networks
Volume 2023, Article ID 2118305, 14 pages
https://doi.org/10.1155/2023/2118305

Research Article

VulDistilBERT: A CPS Vulnerability Severity Prediction Method Based on Distillation Model

Shaofeng Kai, Fan Shi, and Jinghua Zheng

College of Electronic Engineering, National University of Defense Technology, Hefei 230037, China

Correspondence should be addressed to Fan Shi; shifan17@nudt.edu.cn

Received 1 September 2022; Revised 11 October 2022; Accepted 24 November 2022; Published 13 April 2023

Academic Editor: Yuanyuan Huang

With the vigorous development of the Internet, the ecosystem of cyber-physical systems is also developing at a high speed, but cyber-physical systems may be accompanied by unknown vulnerabilities in the process of concrete implementation. Thus, the number of vulnerabilities in cyber-physical systems has been increasing year by year. The vulnerability evaluation speed cannot keep up with the vulnerability exposure speed. The traditional manual evaluation method can no longer effectively deal with such large-scale vulnerabilities, resulting in a backlog of vulnerabilities. Therefore, the vulnerability evaluation results have a certain lag. To address this problem, the paper proposes a vulnerability severity assessment method based on the distillation model. The method first uses data augmentation and integration of optimal subsets to improve the amount of information in the vulnerability description text, then uses the DistilBERT model to characterize the text of the vulnerability description text, and then the characterized feature vectors are classified based on the linear layer to achieve the purpose of assessing vulnerability severity. Compared with the current method of manual assessment based on the CVSS metric system, this method can automate the assessment of vulnerabilities based on vulnerability description text, which improves the speed of vulnerability assessment, and the assessment accuracy and other metrics achieved by this method are improved compared with similar studies. This approach provides an automated solution for cyber-physical systems vulnerability assessment and can better address the current situation where cyber-physical systems vulnerabilities are being exposed at an accelerated rate.

1. Introduction

Cyber-physical systems (CPS) is a set of physical devices (hardware) controlled by computer algorithms (mostly software), and CPS are becoming increasingly important in the information society. However, with high heterogeneity, large-scale deployment, and high dependency on private and sensitive data, CPS is more vulnerable to threats from the network. In recent years, there have been many cases of attacks on CPS hardware, software, and data in the system using vulnerabilities in the CPS, so CPS security needs to be given high priority. This paper provides a priority basis for CPS to repair vulnerabilities by evaluating the severity of CPS vulnerabilities, so as to improve CPS security.

CPS vulnerabilities are flaws in specific implementations of hardware, software, protocols, etc., or system security policies that can enable an attacker to access or compromise a system without authorization. Severity assessment of vulnerabilities can be better for vulnerability severity rating so that according to the severity rating of vulnerabilities to prioritize those high-risk vulnerabilities that have a serious threat to the system, this operation can save the cost and time of vulnerability repair. According to the U.S. National Vulnerability Database (NVD) data, vulnerabilities are showing a trend of increasing year by year, as shown in Figure1. As of September 1, 2022, the NVD has accepted 194,532 vulnerabilities in total. Since the beginning of this year, a total of 16,506 vulnerabilities have been accepted, and an average of nearly 70 vulnerabilities is accepted every day [2]. In addition to the need for vulnerability assessors to assess the severity of newly-added vulnerabilities, some vulnerability exposed earlier due to changes in environmental factors, such as the increase in the number of patch installations, will also lead to changes in the severity of these

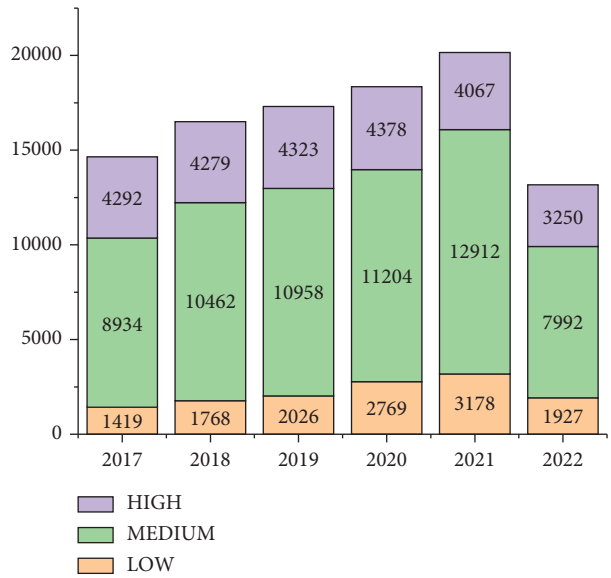


Figure 1: Vulnerability growth trend from January 1, 2017, to September 1, 2022. The data comes from the U.S. National Vulnerability Database [1].

vulnerabilities. The vulnerability assessors also need to re-evaluate this part of the vulnerability. More vulnerability and fewer vulnerability assessors present a sharp contrast; in the long run; vulnerabilities will inevitably appear a backlog phenomenon. This leads to some vulnerability to determine the severity of the time lag behind the time of vulnerability exposure. Research by Chen et al. [3] and Ruohonen [4] show that the vulnerability assessors assess the severity of vulnerability lag behind the time of vulnerability exposure more than 130 days. During this time, unscrupulous individuals may be able to exploit these vulnerabilities to create threats and hazards to the system. There is an urgent need for a method that can quickly assess the severity of vulnerabilities, both to reduce the workload of vulnerability assessors and to improve network security. With the current methods of vulnerability assessment, the speed of vulnerability assessment can be improved by increasing the number of assessors, but this approach will increase the cost for security organizations, so a better way is to propose a method that can automatically assess vulnerabilities.

One idea is to characterize and classify vulnerability description text based on traditional machine learning methods, and some studies also deepen the model feature extraction capability based on deep learning algorithms, but this part of the research does not address the problem of multiple meanings of words and specialized terminology in the cybersecurity domain because algorithms using, for example, TF-IDF algorithms [5], rule-based statistical methods [6], and word embedding [7], cannot capture the contextual word relevance. For example, apple can mean both an apple in fruit and Apple Inc. However, the specific meaning of apple can be known if the context is combined. There are also some studies based on large models, such as BERT [8], but the parameters of models such as BERT are calculated in billions, and such a large model is bound to have an impact on the speed of the training and evaluation process. At the same time, the vulnerability description text also has a small amount of text information, creating the problem of data imbalance. According to statistics, the average description text contained in each vulnerability is about 40 words, and the NVD data show that the severity of the vulnerability there is also a serious imbalance, high-risk and medium-risk vulnerabilities occupy the vast majority of the sample, while low-risk vulnerabilities and critical-risk vulnerabilities occupy only a small part, if this part of the data used directly for training, it will inevitably lead to model bias and instability.

To address the above problems, this paper proposes a vulnerability severity assessment method based on the distillation model, which uses DistilBERT model to characterize the vulnerability description text, because DistilBERT model has been pretrained based on the general corpus, and the model itself already contains more knowledge, which can better extract the dependency relationship between features based on contextual information, thus solves the problem of multiple meanings of words and specialized terms, and then the features are further extracted using the LSTM [9] and CNN [10] algorithm, because for text classification tasks, the sentiment polarity of sentences often consists of individual words or phrases, and the positions of these decisive words and phrases in the sentences are not fixed, and it is difficult to capture such critical local information directly using the fully connected layer, and the data will be passed through the model in the process of the LSTM network can better solve the above problems. The article also addresses the problem of low data information and data imbalance and uses methods such as the data augmentation algorithm and optimal subset integration to describe the text for a better solution.

The experiments in this paper also validate the effectiveness of our proposed method. Compared with the current SOTA method, the accuracy of our evaluation is 96.62%, while the accuracy of SOTA is 93.95%, which is an improvement of 2.67%.

To address the challenge of vulnerability assessment, VulDistilBERT is proposed. The article has the following three main contributions:

(i) In this paper, we use the distillation model to better solve the problem that the text features are not strongly dependent and the word multiple meanings and technical terms cannot be recognized well.

(ii) This paper compiles a vulnerability assessment dataset based on NVD data and solves the problems of insufficient information and data imbalance based on data augmentation techniques and the optimal subset integration.

(iii) This paper proposes a vulnerability severity assessment framework based on the distillation model, which can automate the assessment of vulnerability severity levels. This method increases the speed of evaluation and reduces the workload of manual evaluation.

2. Background

2.1. The Common Vulnerability Scoring System. The Common Vulnerability Scoring System (CVSS) is an open framework developed by FIRST.Org, Inc. (FIRST) to characterize and quantify vulnerabilities. CVSS consists of three metric groups: the base metric group, the temporal metric group, and the environmental metric group. Since major vulnerability databases only provide base scores, the base metric group is the most used. The focus of this study is the base metric group. The base metric group reflects the inherent properties of vulnerabilities that remain unchanged over time and across user environments. The base metric group generates scores ranging from 0 to 10. The CVSS score can also be shown as a vector string, which is a textual way to represent the metric value compactly. The NVD, when using the CVSS, usually gives a string representation of the description and the corresponding vulnerability metric values. Table 1 shows the eight metrics and their meanings contained in the base metric group of CVSS V3.1, as well as the possible values of the metrics. The metric values are substituted into the CVSS quantification formula to obtain the base score. Finally, the base score can be converted into a vulnerability rating [12–14].

2.2. DistilBERT Model

2.2.1. The Meaning of Knowledge Distillation. Unlike pruning and quantization in model compression, knowledge distillation is a common method of model compression and knowledge migration. Knowledge distillation is done by constructing a small lightweight model and using the supervised information of a larger model with better performance to train the small model to achieve better performance and accuracy. As the name suggests, the teacher is responsible for teaching knowledge, and the student is responsible for receiving and digesting it. The final goal is that the student has successfully transferred most of the knowledge from the teacher's brain to his or her own brain, completing the knowledge distillation [15].

2.2.2. Purpose of Knowledge Distillation. It is not easy for humans to learn a subject well, let alone a model with no feelings and no brain. The only thing it can do is to use its big computing power to solve the problem; maybe a human can distinguish the emotional color of a sentence at a glance, while the machine will need constant trial and error corrections to achieve a closer result with humans. Therefore, this also leads to the large parameters of large models such as BERT. The parameters of the basic BERT model are about 115 million; however, this is still the only entry-level parameter in language models, which brings challenges for model training, evaluation, deployment, and application. If we can achieve or approach the effect that can only be achieved by the raw large model with a lightweight model, it can allow us to achieve two things in one. Experimental results demonstrate that DistilBERT reduces the model size by 40% and the inference operations are 60% faster while retaining 97% of the performance [16].

2.2.3. The Process of Knowledge Distillation. BERT is mainly based on a series of attention layers stacked on top of each other, so this means that the "hidden knowledge" learned by BERT is contained in these layers. The major difference between BERT models is the number of layers N, which is naturally proportional to the size of the model. It follows that the time taken to train the model and the time for forwarding propagation also depends on N and, of course, the memory used to store the model. Therefore, the logical conclusion of distillation BERT is to reduce N. DistilBERT is done by halving the number of layers and initializing the student's layer from the teacher's layer. For example, a distillation of the base BERT is done by distilling the 12-layer BERT to obtain a 6-layer DistilBERT, first pretraining the BERT, and then distilling the 12-layer BERT with a large-scale prefeed of the training BERT to obtain it. The knowledge distillation process is shown in Figure 2.

3. Methods

This study aims to design an efficient approach using vulnerability description text to predict vulnerability severity. This approach will help analysts quickly analyze the vulnerability severity levels. The paper proposes a learning method based on the DistilBERT model, which fine-tunes the DistilBERT model to improve the model's learning efficiency and prediction effect.

3.1. Methodology Overview. Figure 3 depicts the paper's two primary phases: The DistilBERT model transfer learning and severity prediction. By employing DistilBERT model transfer learning, a fine-tuned model will be developed to predict vulnerability severity. Severity prediction is a four-step process, i.e., data integration, text tokenization, token embedding, and vulnerability severity prediction. These steps are shown in Figure 3. First, data augmentation and optimal subset integration operations are performed on the raw data, and the integrated data is used as model input, then the vulnerability description content is divided into numerous tokens during the tokenization stage, and then the tokens are embedded by the fine-tuned DistilBERT model. Finally, the paper uses the linear function to predict the severity of vulnerabilities. The remainder of this section will provide a detailed description of the framework.

3.2. DistilBERT Transfer Learning. DistilBERT transfer learning fine-tunes the pretrained model utilizing the self-built corpus collected in the NVD. The pretrained DistilBERT is obtained by distilling the Bert model. DistilBERT transfer learning begins with downloading the appropriate pretrained DistilBERT model. In this study, the pretrained model is "DistilBERT-base-uncased." Then, the domain corpus is used to fine-tune the model. The domain corpus is constructed from 1999 to 2022 NVD vulnerability descriptions. The input and output relationships of the DistilBERT model are shown in the following equation:

$$\mathbf{e}^{[l]} = f_{\text{DistilBERT}}\left(\Theta_{pre-\text{DistilBERT}}, \mathbf{t}\right) \tag{1}$$

Table 1: Description and possible values for base metric group of CVSS. The metric names in this table will be referred to by abbreviations and ID number in [11].

ID	Metric	Description	Possible values
0	Attack vector (AV)	This metric represents the conditions under which exploiting vulnerability is conceivable. The farther an attacker may exploit a susceptible component, the higher this metric	Physical Network Local Adjacent
1	Attack complexity (AC)	This metric reflects the attacker-uncontrolled circumstances needed to exploit the vulnerability	Low High
2	Privileges required (PR)	This metric represents an attacker's privilege before exploitation. The score is the highest when no privileges are necessary	None Low High
3	User interaction (UI)	This metric represents the necessity for a human user other than the attacker to be involved in the successful penetration of the susceptible component	Required None
4	Scope (S)	This metric measures whether one component's vulnerability impacts other components' resources	Unchanged Changed
5	Confidentiality (C)	This metric quantifies the confidentiality of a successfully exploited vulnerability on the component most directly and predictably affected by the attack	None Low High
6	Integrity (I)	This metric represents vulnerability's influence on integrity. Integrity means truthfulness and trustworthiness	None Low High
7	Availability (A)	This metric assesses a vulnerability's influence on a component's availability	None Low High

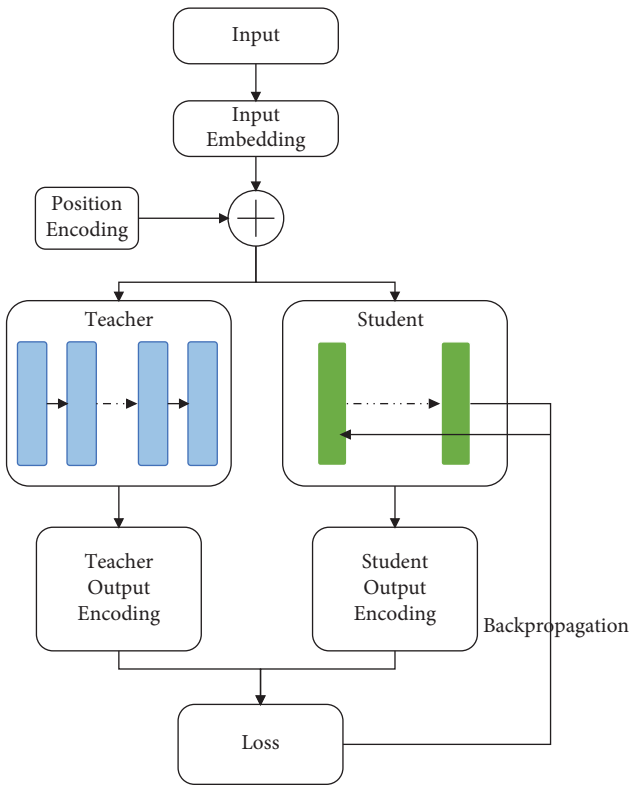


Figure 2: Distillation model working process. Students gain knowledge by learning from teachers.

where $\mathbf{t} = [t_1, t_2, \cdots, t_n]$ is a sequence of the token list with n tokens, which are tokenized based on the vulnerability description text; $\mathbf{e}^{[l]} = [e_1^{[l]}, e_2^{[l]}, \cdots, e_n^{[l]}]$ is the pretrained DistilBERT model's l th layer token embedding; $\Theta_{pre-\text{DistilBERT}}$ represents the parameters of the pretrained DistilBERT model; $f_{\text{DistilBERT}}(\cdot)$ is the conversion function of $\mathbf{t}$ and $\mathbf{e}^{[l]}$, determined by the DistilBERT structure; $e_i^{[l]}$ is the l th layer token embedding of the i th token t_i, $e_i^{[l]} \in \mathbb{R}^{H_{\text{DistilBERT}}^{[l]}}$, where $H_{\text{DistilBERT}}^{[l]}$ is the DistilBERT l th layer hidden layer size. By transfer learning, the parameters of DistilBERT are changed from its pretrained state $\Theta_{\text{pre-DistilBERT}}$ to its fine-tuned state $\Theta_{\text{fine-tuned DistilBERT}}$. Compared to training a DistilBERT model from scratch, using transfer learning on a DistilBERT model maintains a comprehensive model's high performance while avoiding the high training cost and lack of domain data [17].

3.3. Vulnerability Severity Prediction

3.3.1. Data Integration

(1) Data Augmentation. The statistics of the vulnerability description text showed that there was a serious imbalance in the data labels, and the results are shown in Figure 4. CRITICAL, HIGH, MEDIUM, and LOW accounted for 15.1%, 42.9%, 40.1%, and 1.9%, respectively. The low percentage of low-risk vulnerabilities may cause the trained model to fail to accurately predict low-risk vulnerabilities, which also affects the robustness of the model. In this paper, we adopt the method of data augmentation to solve the data imbalance problem. Data augmentation (DA) is a representation of processing more data from the raw data without substantially increasing the data and improving the quantity and quality of the raw data to approach the value generated by more data volume to improve the learning effect of the model. In this paper, we mainly adopt synonym replacement (SR) and random deletion (RD) to expand the sample with a small number of tags [18]. Synonym replacement refers to randomly selecting n words from the sentence that do not belong to the deactivated word set and randomly choosing their synonyms to replace them. Random deletion refers to randomly removing each word in a sentence with a probability of p.

(2) Optimal Subset. The analysis of the vulnerability description text shows that the average length of each vulnerability description statement is 43.77 words and 82.96% of the vulnerability description statements are less than 64 words, which is shown in Figures 5 and 6. When the model is trained to a certain level, the prediction effect of the model can no longer be improved, and even overfitting will occur.

In this paper, we adopt the method of extracting CVSS metrics and selecting the optimal subset (OS) of metrics to be incorporated into the vulnerability description text to improve the amount of textual information. The current latest version of CVSS is 3.1, whose base metric group contains 8 metrics, and we investigate the impact of these 8 metrics on the final vulnerability severity classification. In this study, CVSS metrics are first combined according to the

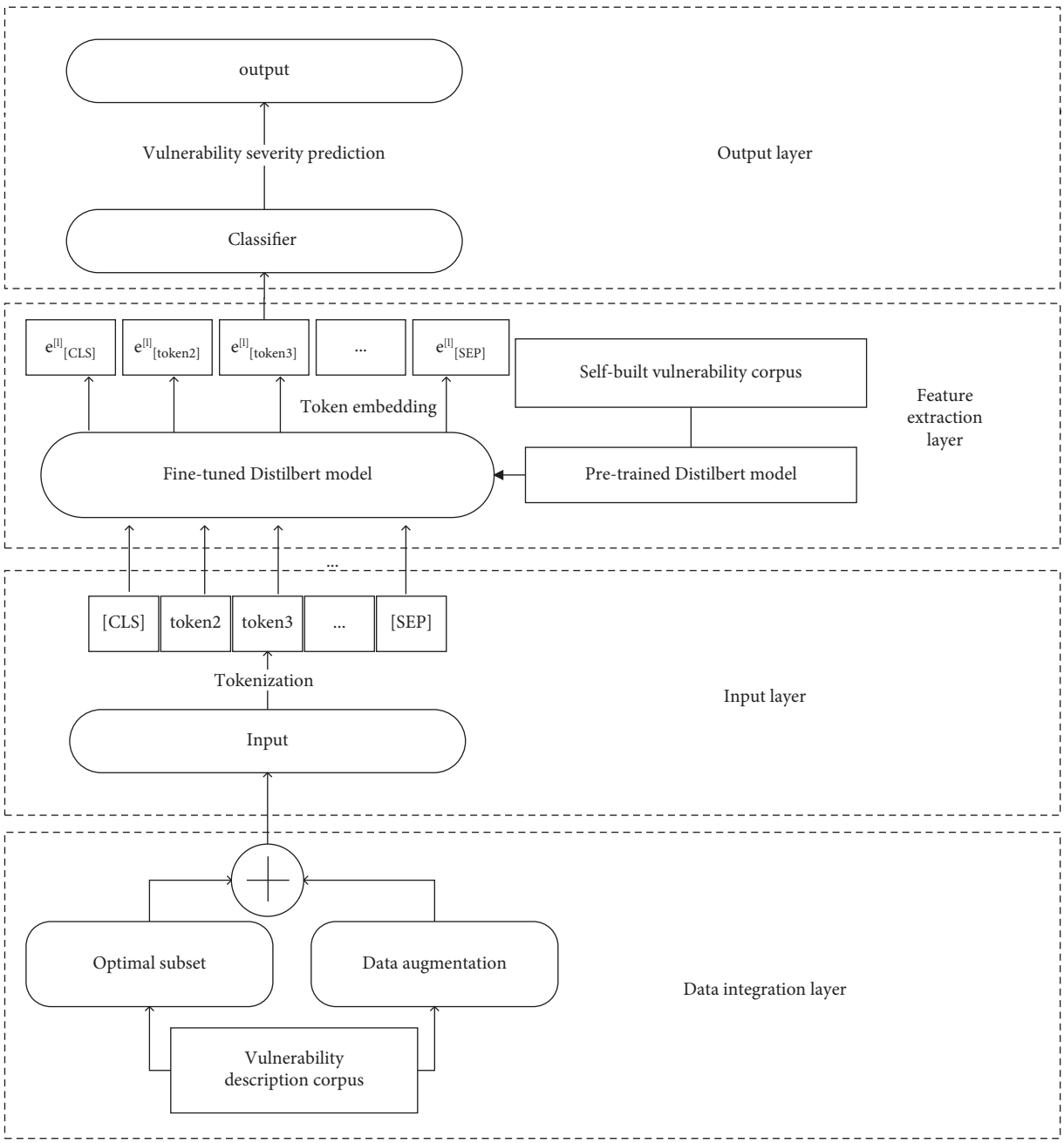


Figure 3: The framework structure of the algorithm in this paper. The framework mainly includes the data integration layer, input layer, feature extraction layer, and output layer.

full ranking method, and then a variety of typical machine learning methods are selected to select the optimal subset [19], and the relevant methods are random forest, decision tree, KNN, AdaBoost, SVM, logistic regression, multilayer perceptron, gradient boosting algorithm, and so on. Because the optimal subset is selected to improve the final vulnerability severity level classification effect, finally this paper lists the classification effect of related machine learning algorithms on the combination of metrics, due to the reason of space, only the first two combinations with better effect in each category are listed and the combination with the best effect is shown in bold. The results are shown in Tables 2 and 3. The numbers in the table refer to the metric names; please refer to Table 1 for the specific reference relationship. The effects of the algorithms are then combined to finally determine the optimal subset of each class of combinations, as shown in Table 4.

3.3.2. Model Building

(1) Input Layer. The main function of the input layer is text tokenization. Text tokenization is a data preprocessing process in which the description text $X = [V_1, V_2 \cdots, V_n]$ are turned into token sequences $T = [t^{(1)}, t^{(2)}, \cdots, t^{(n)}]$, $\mathbf{t}^{(i)} = [t_1, t_2, \cdots, t_k]$ is the token sequences obtained from the

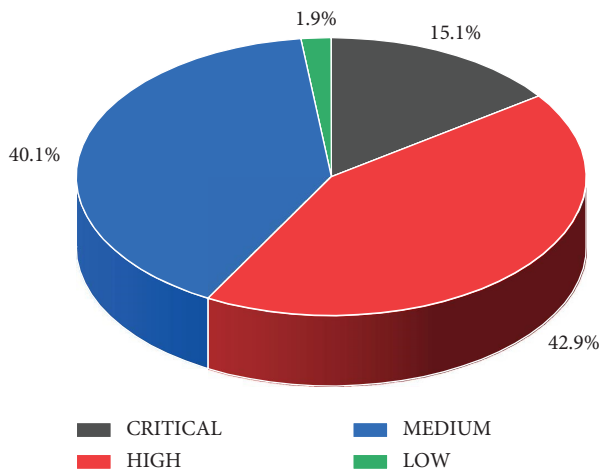


Figure 4: The proportion of each category of raw data.

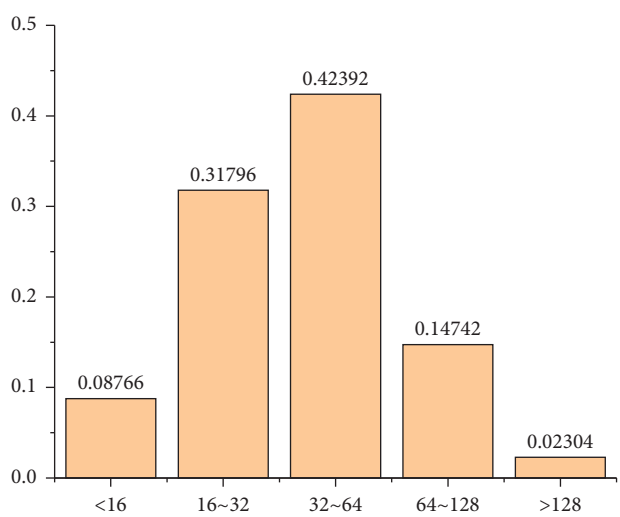


Figure 5: The distribution of the number of words in the vulnerability description text. The maximum number of words is in the range of 32–64.

description V_i; k is the maximum sequence length of the preset token. The symbol t_j denotes the j th token obtained from the characterization of the description text V_i, $i \in \{1, 2, \cdots, n\}$, $j \in \{1, 2, \cdots, k\}$.

(2) Feature Extraction Layer. When text tokenization is done, the result is what goes into token embedding. When the fine-tuned DistilBERT is given a token list $\mathbf{t}$, different transfer layers will give different levels of token embedding. For instance, this is how the token embedding of the layer l from the fine-tuned DistilBERT is shown as follows:

$$\mathbf{e}^{[\mathbf{l}]} = f_{\text{DistilBERT}}\left(\Theta_{\text{fine-tuned DistilBERT}}, \mathbf{t}\right) \tag{2}$$

Similar to (1), $\mathbf{t} = [t_1, t_2, \cdots, t_k]$ is a token sequence that consists of k tokens, $\mathbf{e}^{[l]} = [e_1^{[l]}, e_2^{[l]}, \cdots, e_n^{[l]}]$ represents the l th layer of token embedding. $e_i^{[l]}$ is the i th token t_i, $e_i^{[l]} \in \mathbb{R}^{H_{\text{DistilBERT}}^{[l]}}$, where $H_{\text{DistilBERT}}^{[l]}$ is the DistilBERT l th layer hidden size.

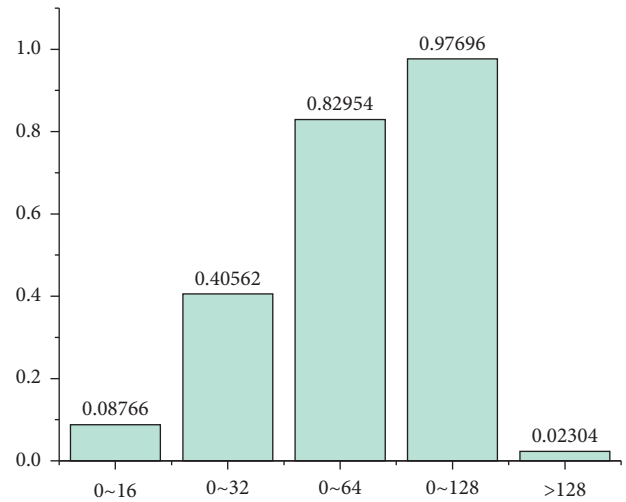


Figure 6: Cumulative distribution of the number of words in the vulnerability description text. Vulnerability description text less than 128 words accounted for the vast majority.

(3) Output Layer. In the research, we utilize the fully connected layer as a classifier. It can perform a linear transformation on the incoming data. The following is the formula to use: $\hat{y}_i = W^k \mathbf{e}^{[\mathbf{L}]} + b^k$, $\hat{y}_i$ is the possible category of the vulnerability severity, W^k and b^k are the functions' weights and biases, $\mathbf{e}^{[\mathbf{L}]}$ is the token embedding of the final layer output.

4. Experiments and Results

4.1. Experimental Data and Experimental Setup. The paper uses data from the US National Security Vulnerability Database [20], which contains all security vulnerabilities released from 1999 to 2022. The vulnerability description information in the web page is used as the dataset's text item, and the severity of vulnerabilities is processed to label items. There are four types of labels in the dataset, which correspond to the severity of four types of vulnerabilities, i.e., critical risk, high risk, medium risk, and low risk. The collected dataset is represented as $D = \{X, Y\}$. Examples of datasets are shown in the following Table 5. The vulnerability description item in the table is the input of the model, and the severity item is the label of the model. After processing, the dataset contains 105,984 vulnerability samples. The dataset was split into the training and test datasets in the following proportions: 85%: 15%. The statistics indicate that 99.8% of vulnerability descriptions are less than 256 words, with an average of 43.77 words per sample. The pretrained DistilBERT model used in the paper is the "DistilBERT-base-uncased" model [21]. In the data integration phase, the value of SR is set to 0.05 and the value of RD is set to 0.1. In the integration of the optimal subset, the integration is the optimal subset composed of four metrics: PR, UI, C, and I. All vulnerability descriptions were used to fine-tune the pretrained DistilBERT model. Two NVIDIA GeForce RTX 3090 GPUs were used for fine-tuning and training.

Table 2: Random forest, decision tree, AdaBoost, and KNN algorithms for evaluation results of metric combinations. Taking the first row of the random forest algorithm as an example, the vulnerability severity assessment using only one metric, 5, can still achieve an accuracy rate of 65.39%.

Random forest		Decision tree		AdaBoost		KNN	
Metric combination	Accuracy	Metric combination	Accuracy	Metric combination	Accuracy	Metric combination	Accuracy
5	0.6539	5	0.6539	5	0.6539	5	**0.6539**
6	**0.6687**	6	**0.6687**	6	**0.6687**	7	0.6523
0, 6	0.6749	0, 6	0.6749	0, 6	0.6749	0, 7	**0.6581**
5, 6	**0.6765**	5, 6	**0.6765**	5, 6	**0.6765**	3, 5	0.6578
2, 3, 5	0.7785	2, 3, 5	0.7785	2, 3, 5	**0.7785**	0, 2, 6	0.7637
2, 3, 6	**0.8238**	2, 3, 6	**0.8238**	2, 3, 6	0.7694	2, 3, 6	**0.8239**
2, 3, 5, 6	**0.8894**	2, 3, 5, 6	**0.8894**	2, 3, 5, 6	**0.8882**	2, 3, 5, 6	**0.8804**
2, 3, 5, 7	0.8841	2, 3, 5, 7	0.8841	2, 3, 5, 7	0.8841	2, 3, 5, 7	0.8646
0, 2, 3, 5, 6	**0.9280**	0, 2, 3, 5, 6	**0.9281**	0, 2, 3, 5, 7	**0.9225**	0, 2, 3, 5, 6	**0.9262**
0, 2, 3, 5, 7	0.9225	0, 2, 3, 5, 7	0.9225	1, 2, 3, 5, 7	0.9123	0, 2, 3, 5, 7	0.9219
0, 1, 2, 3, 5, 6	**0.9574**	0, 1, 2, 3, 5, 6	**0.9575**	0, 1, 2, 3, 5, 6	**0.9527**	0, 1, 2, 3, 5, 6	**0.9560**
0, 2, 3, 5, 6, 7	0.9525	0, 2, 3, 5, 6, 7	0.9523	0, 1, 2, 3, 5, 7	0.9480	0, 2, 3, 5, 6, 7	0.9493
0, 1, 2, 3, 4, 5, 6	0.9699	0, 1, 2, 3, 4, 5, 6	0.9699	0, 1, 2, 3, 5, 6, 7	**0.9850**	0, 1, 2, 3, 4, 5, 6	0.9685
0, 1, 2, 3, 5, 6, 7	**0.9852**	0, 1, 2, 3, 5, 6, 7	**0.9849**	0, 2, 3, 4, 5, 6, 7	0.9640	0, 1, 2, 3, 5, 6, 7	**0.9844**

Table 3: SVM, logistic regression, multilayer perceptron, and gradient boosting algorithms for evaluation results of metric combinations.

SVM		Logistic regression		Multilayer perceptron		Gradient boosting	
Metric combination	Accuracy	Metric combination	Accuracy	Metric combination	Accuracy	Metric combination	Accuracy
5	0.6539	5	0.6539	5	0.6539	5	0.6539
6	**0.6687**	6	**0.6687**	6	**0.6687**	6	**0.6687**
0, 6	0.6749	5, 6	**0.6765**	0, 6	0.6748	0, 6	0.6749
5, 6	**0.6765**	5, 7	0.6701	5, 6	**0.6765**	5, 6	**0.6765**
2, 3, 5	0.7785	2, 3, 6	**0.7976**	0, 3, 6	0.7352	2, 3, 5	0.7785
2, 3, 6	**0.8238**	2, 5, 7	0.7093	2, 3, 5	**0.7774**	2, 3, 6	**0.8238**
2, 3, 5, 6	**0.8892**	2, 3, 5, 7	**0.87412**	2, 3, 5, 7	**0.8823**	2, 3, 5, 6	**0.8887**
2, 3, 5, 7	0.8839	2, 3, 6, 7	0.8275	2, 3, 6, 7	0.8461	2, 3, 5, 7	0.8810
0, 2, 3, 5, 6	**0.9274**	1, 2, 3, 5, 7	**0.8962**	0, 2, 3, 5, 7	**0.9123**	0, 2, 3, 5, 6	**0.9244**
0, 2, 3, 5, 7	0.9218	2, 3, 5, 6, 7	0.8935	1, 2, 3, 5, 6	0.9069	1, 2, 3, 5, 6	0.9187
0, 1, 2, 3, 5, 6	**0.9575**	0, 2, 3, 5, 6, 7	0.9130	0, 1, 2, 3, 5, 7	0.9265	0, 1, 2, 3, 5, 6	0.9443
0, 2, 3, 5, 6, 7	0.9522	1, 2, 3, 5, 6, 7	**0.9271**	1, 2, 3, 5, 6, 7	**0.9424**	0, 2, 3, 5, 6, 7	**0.9488**
0, 1, 2, 3, 4, 5, 6	0.9691	0, 1, 2, 3, 5, 6, 7	**0.9486**	0, 1, 2, 3, 5, 6, 7	**0.9616**	0, 1, 2, 3, 5, 6, 7	**0.9792**
0, 1, 2, 3, 5, 6, 7	**0.9845**	1, 2, 3, 4, 5, 6, 7	0.9252	1, 2, 3, 4, 5, 6, 7	0.9470	0, 2, 3, 4, 5, 6, 7	0.9544

Table 4: The optimal subset selected for each combination.

Optimal subsets	Included metrics
Optimal subset with 1 metric	I
Optimal subset with 2 metrics	C, I
Optimal subset with 3 metrics	PR, UI, I
Optimal subset with 4 metrics	PR, UI, C, I
Optimal subset with 5 metrics	AV, PR, UI, C, I
Optimal subset with 6 metrics	AV, AC, PR, UI, C, I
Optimal subset with 7 metrics	AV, AC, PR, UI, C, I, A

4.2. Ablation Experiments. In this section, we set up multiple groups of control experiments to compare the classification effects of several pretrained models in fine-tuned and non-fine-tuned states, set up the effects of average pooling, maximum pooling, and no pooling on the classification results, and compare the differences in the effects of the linear layer, CNN, and LSTM methods as classifiers. It is demonstrated that the fine-tuned DistilBERT has a smaller size, faster inference speed, and top prediction results compared to other pretrained models.

4.2.1. Pretrained Model Effect Analysis. Pretrained models are a class of deep learning frameworks that have been trained on a large amount of data and can usually be used directly for downstream tasks, but since the data distribution at the time of pretraining may not be the same as that of the downstream tasks, the models may not be well adapted to the downstream tasks, so when pretrained models are used, they are usually fine-tuned on the existing data. In this study, four pretrained models were selected for the vulnerability assessment task, and their vulnerability classification effects were compared in the fine-tuned and non-fine-tuned states, respectively. The four pretrained models are BERT, XLNET [22], ROBERTA [23], and DistilBERT. The results are shown in the following table. Table 6 shows the effect of the test set on each pretrained model and Table 7 shows the effect of the test set on each fine-tuned model with the best number of

Table 5: Sample example of part of the dataset.

CVE ID	Vulnerability description	Severity
CVE-2022-40980	Potential unauthenticated file deletion vulnerability on Trend Micro Mobile Security for Enterprise 9.8 SP5 could allow an attacker with access to the Management Server to delete files. This issue was resolved in 9.8 SP5 Critical Patch 2	Critical
CVE-2022-3079	Festo control block CPX-CEC-C1 and CPX-CMXX in multiple versions allow unauthenticated, remote access to critical webpage functions which may cause a denial of service	High
CVE-2022-38846	EspoCRM version 7.1.8 is vulnerable to missing secure flag allowing the browser to send plain text cookies over an insecure channel (HTTP). An attacker may capture the cookie from the insecure channel using MITM attack	Medium
CVE-2022-39850	Improper access control in mum_container_policy service prior to SMR Oct-2022 release 1 allows allows unauthorized read of configuration data	Low

epochs taken as the number of epochs with the smallest validation loss. From the data in the table below, it can be seen that the DistilBERT model performs better on the test set in terms of accuracy and precision before fine-tuning, and requires fewer training epochs, while the performance after fine-tuning is generally on par with other models and exceeds them in some metrics. Table 8 gives a comparison of the performances.

4.2.2. Pooling Effect Analysis. The pooling layer can further aggregate the upstream features, which can effectively reduce the size of the parameter matrix, thereby reducing the number of parameters in the final connection layer, so adding a pooling layer can speed up the computation and prevent overfitting. We analyze the pooling effect for the fine-tuned DistilBERT models with maximum pooling, minimum pooling, and no pooling, respectively [24]. The results are shown in Table 9. From the data in the following table, it can be seen that the final classification effect is not much improved for the vectors obtained from the pretrained model and then pooled, and it is even not as good as the effect without pooling.

4.2.3. Data Augmentation Effect Analysis. In this paper, the vulnerability descriptions labeled low were amplified by 15 times, and the vulnerability descriptions labeled critical were amplified by 2 times, while the number of other category labels remained unchanged. The raw data sample size is 105,984, and after amplification, the sample size is 150,159. The percentage of each category is shown in Figure 7. To evaluate the impact and effect of augmented data, the integrated optimal subset of data, and both data augmentation and the integrated optimal subset of data on vulnerability severity prediction, we determined the token embedding model as a fine-tuned DistilBERT without pooling. The data is also validated using different classifiers under the same dataset. Table 10 gives the evaluation results of different classifiers in the vulnerability description text with data augmentation; Table 11 gives the evaluation results of different classifiers in the vulnerability description text with integrated optimal subset; and Table 12 gives the evaluation results of different classifiers in the vulnerability description text with both data augmentation and integrated optimal subset. Analysis of the data in the table below shows that the vulnerability description text with data augmentation improves the model better than the vulnerability description text incorporating the optimal subset. In the vulnerability description text with data augmentation, the accuracy of the pretrained model improves by 9% over the raw data and the accuracy of the fine-tuned model improves by 14% over the raw data with the same classifier. In the vulnerability description text incorporating the optimal subset, the accuracy of the pretrained model improves by 1% over the raw data, and the accuracy of the fine-tuned model improves 15% over the raw data with the same classifier, while the vulnerability description text with both data augmentation and incorporation of the optimal subset has better results than the single means, the accuracy of the pretrained model improves 19% over the raw data with the same classifier, and the accuracy of the fine-tuned model improves 19% over the raw data with the same classifier. The accuracy of the pretrained model improves by 19% over the raw data, 10% over the data-enhanced text, and 18% over the text incorporating the optimal subset. The accuracy of the fine-tuned model improved by 19% over the raw data, 5% over the data-augmented text, and 4% over the text incorporated into the optimal subset.

4.3. Comparison with Other Similar Works. In the study, the accuracy, precision, recall, and F1 score of fine-tuned DistilBERT are compared with those of similar works. The results of other works are taken from the original paper. Table 13 displays the pertinent data. The results demonstrate that our method enhances the performance of vulnerability severity prediction.

4.4. Analysis of Results. Tables 6 and 7 give the classification effects of different models in fine-tuning and pretraining states using a linear layer on the raw data, combined with training epochs, accuracy, and other data, DistilBERT model is selected as the text characterization model, Table 8 gives the improvement effect of DistilBERT model in the fine-tuning state compared with the pretraining state. From the data, it can be seen that in the fine-tuned model, the metrics are significantly improved, thus proving that the fine-tuning approach does have a significant improvement on the task.

Table 6: Results of evaluating the raw dataset with the pretrained model.

Models	Best epoch	Loss	Accuracy	Precision	Recall	F1 scores
BERT	26	1.038	0.54	**0.42**	0.33	0.32
XLNET	48	**0.978**	**0.56**	0.41	**0.36**	**0.35**
ROBERTA	74	1.043	0.54	0.34	0.32	0.29
DistilBERT	**24**	1.015	**0.56**	**0.42**	0.35	0.34

Table 7: Results of evaluating the raw dataset with the fine-tuned model.

Models	Best epoch	Loss	Accuracy	Precision	Recall	F1 scores
Fine-tuned BERT	**2**	0.580	0.77	0.75	0.62	0.66
Fine-tuned XLNET	4	0.570	0.78	0.74	0.65	0.68
Fine-tuned ROBERTA	4	**0.566**	0.78	0.76	0.65	0.68
Fine-tuned DistilBERT	4	0.571	**0.78**	**0.76**	**0.68**	**0.70**

Table 8: Performance comparison between pretrained model and fine-tuned model.

Models	Pretrained DistilBERT	Fine-tuned DistilBERT	Improvement (%)
Best epoch	24	4	500
Loss	1.015	0.571	43.74
Accuracy	0.56	0.78	28.21
Precision	0.42	0.76	43.59
Recall	0.35	0.68	48.53
F1 score	0.34	0.70	51.43

Table 9: Results of pooling effect comparison using fine-tuned DistilBERT model.

Models	Pool	Best epoch	Loss	Accuracy	Precision	Recall	F1 scores
	Mean	**3**	0.585	0.77	**0.78**	0.64	0.67
Fine-tuned DistilBERT	Max	4	0.588	0.77	0.75	0.64	0.67
	CLS	4	**0.571**	**0.78**	0.76	**0.68**	**0.70**

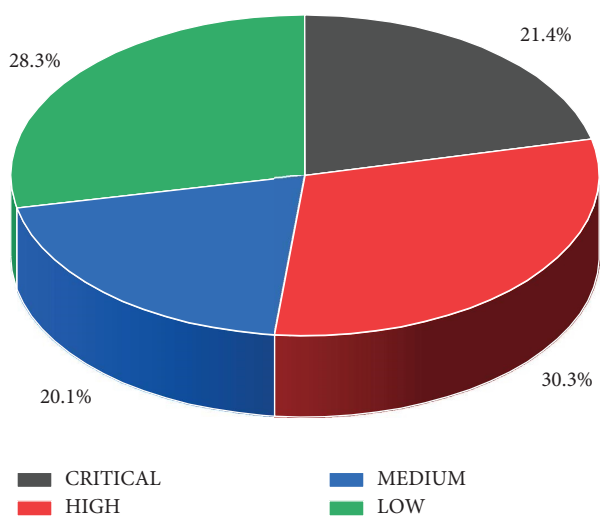


Figure 7: After data augmentation, the proportion of each category in the dataset.

The effects of different pooling effects on the model classification are given in Table 9. From the data in the table, it can be concluded that adding a pooling layer does not improve the model performance well, but increases the complexity of the model. Tables 10–12 give the control results of different information augmentation methods. From the data in the table, it can be analyzed that CNN, LSTM, and other complex networks do not improve the classification effect of the model well, probably because fine-tuned DistilBERT model has already characterized the features of the model well, even if a deeper network is used for feature extraction, its effect will not be better than the simple linear layer effect. Moreover, the data in Tables 10–12 can be analyzed to conclude that performing information

Table 10: Results of comparing the performance of different classifiers on a data-augmented dataset.

Models	Classifiers	Best epoch	Loss	Accuracy	Precision	Recall	F1 scores
Pretrained DistilBERT	Linear	32	0.927	0.65	0.66	0.65	0.65
Fine-tuned DistilBERT	Linear	3	0.221	0.92	0.92	0.92	0.92
Fine-tuned DistilBERT	CNN	4	0.225	0.91	0.92	0.92	0.92
Fine-tuned DistilBERT	LSTM	3	0.228	0.91	0.92	0.92	0.92

Table 11: Results of comparing the performance of different classifiers on the dataset integrated with the optimal subset.

Models	Classifiers	Best epoch	Loss	Accuracy	Precision	Recall	F1 scores
Pretrained DistilBERT	Linear	31	1.035	0.57	0.43	0.34	0.31
Fine-tuned DistilBERT	Linear	4	0.231	0.93	0.88	0.87	0.87
Fine-tuned DistilBERT	CNN	4	0.237	0.93	0.90	0.87	0.88
Fine-tuned DistilBERT	LSTM	4	0.234	0.93	0.86	0.90	0.88

Table 12: The results of comparing the performance of different classifiers on the dataset with the optimal subset integrated and data augmented.

Models	Classifiers	Best epoch	Loss	Accuracy	Precision	Recall	F1 scores
Pretrained DistilBERT	Linear	34	0.814	0.75	0.77	0.76	0.76
Fine-tuned DistilBERT	Linear	2	0.095	0.97	0.97	0.97	0.97
Fine-tuned DistilBERT	CNN	2	0.093	0.97	0.97	0.97	0.97
Fine-tuned DistilBERT	LSTM	2	0.092	0.97	0.97	0.97	0.97

Table 13: Comparison of results related to similar works.

Researchers	Feature extraction method	Classifiers	Accuracy (%)	Precision (%)	Recall (%)	F1 scores (%)
Khazaei et al. [25]	TF-IDF	Fuzzy system	88.37	—	—	—
Spanos et al. [26]	Document-term matrix	Decision tree	79.12	75.54	71.26	73.02
		Neural network	78.26	73.59	70.24	71.68
		SVM	79.53	78.49	68.21	71.50
Nakagawa et al. [27]	One-hot	CNN	72.50	—	—	—
Wang et al. [28]	Feature vector	PCA + XGBOOST	92.38	—	—	—
Han et al. [29]	Word embedding	1-Layer CNN	81.60	81.80	81.50	81.60
Liu et al. [30]	Text mining	XGBoost	87.30	—	—	—
		CNN	92.04	—	—	—
		LSTM	93.73	—	—	—
		TextRCNN	93.95	—	—	—
Our method	Fine-tuned DistilBERT with DA	Linear	90.64	91.92	91.92	91.83
	Fine-tuned DistilBERT with OS	Linear	92.80	88.32	87.26	87.07
	Fine-tuned DistilBERT with DA and OS	Linear	96.62	97.11	97.06	97.05

augmentation does have a better enhancement effect on the model compared to the raw dataset, where the dataset combining data augmentation and optimal subset integration has the best effect, reaching 97% accuracy, ahead of other current methods.

4.5. Discussion. The experimental results show that the fine-tuned DistilBERT model does have a better prediction capability for vulnerability severity, which benefits from both the power of the pretrained model itself and the knowledge provided by the fine-tuned corpus. Compared with traditional machine learning and deep learning, the DistilBERT model obtains a large amount of general knowledge from the large-scale corpus and domain knowledge from the fine-tuned corpus, which is the key to the successful application of the large model to downstream tasks. This knowledge compensates for the difficulties caused by insufficient data in downstream tasks, thus significantly improving them. In addition, the results show that pooling the fine-tuned DistilBERT model followed by a pooling layer and classification followed by a deep learning model do not further improve the model, which also indicates that the DistilBERT model has already performed a good feature extraction of the data, and even adding other modules cannot optimize the features further, but rather increases the complexity of the model, which is not conducive to the application of the model. After the optimal subset integration and data enhancement, the model effect has been improved. The key is that the data enhancement technique generates new data by perturbing the data itself to a certain extent, and the model improves its generalization ability by continuously learning

a large amount of new data, and the optimal subset also further increases the information content of the data. However, the current model has problems, such as large number of parameters and low interpretability. In future research, we will further study the optimization and interpretability of the model in order to make the model have better application capability.

5. Related Work

Vulnerability assessment is one of the research topics in the field of cyberspace security, which is centered on threat assessment, risk level assessment, and vulnerability rating score for vulnerabilities. Current research on vulnerability assessment is mainly based on quantitative research and model-based research.

The CVSS is the de-facto vulnerability assessment standard, and many studies have been conducted based on it. Spanos et al. [31] has adjusted the weights and proposed a method called the Weighted Impact Vulnerability Scoring System (WIVSS). Based on this, Luo et al. [32] analyzed that CVSS cannot distinguish software vulnerabilities with the same score but different severity levels. A software vulnerability rating method, SVRA, was proposed, which uses a vulnerability database to analyze the frequency of CVSS metrics at different times, and then gives equations for exploitability and impact scores based on these frequencies. Wang et al. [33] considered that CVSS metrics ignore the impact of vulnerabilities on specific networks, i.e., the same vulnerabilities that exist in different network environments are assigned duplicate values, and the attack graph still suffers from scalability and readability issues. To address the above issues, the authors innovatively propose a vulnerability risk assessment method based on heterogeneous information networks. This method takes into account the exploitability of vulnerabilities, the impact of vulnerabilities on network components, and the importance of vulnerable components. The above three articles represent three directions of such research, i.e., focusing on the inadequacy of CVSS evaluation weights, the inadequacy of CVSS evaluation methods, and the inadequacy of CVSS evaluation metrics, but the above methods are mostly based on quantitative formulas to assess vulnerabilities, and the metric values required for the assessment process still need to be extracted manually, which does not significantly improve the speed of hole assessment.

The model-based approach, on the other hand, hopes to take advantage of machine learning algorithms to automatically extract features through algorithms to avoid the bias brought about by manual extraction of features, and the main idea is to characterize and classify vulnerability description statements based on machine learning methods. Khazaei et al. [25], Wang et al. [28], Han et al. [29], and Liu et al. [30] have characterized vulnerability descriptions using traditional word-vector algorithms. However, such methods do not consider contextual information, which may contain rich information that could enhance the final prediction. Based on these studies, Spanos et al. [26], Ali [34], Ameri et al. [35], and Kudjo et al. [36] apply traditional machine learning algorithms and deep learning algorithms to CVSS score prediction. The introduction of deep learning algorithms has improved feature extraction to a certain extent. These methods bring convenience to CVSS assessment work. Shahid and Debar [37], Gong et al. [38], and Costa et al. [39] applied pretrained models and deep learning algorithms to vulnerability metric prediction work. However, these methods do not give the final vulnerability severity level prediction. This can be used as a basis for more in-depth research.

At present, research on vulnerability assessment still has common problems to be studied, such as the unscientific establishment of the metric system, inadequate mining of text dependencies, complex model structure, etc. With the development of deep learning, some problems have been solved, but with the development of the network, some new problems have emerged in front of researchers, such as the problem of vulnerability hazard assessment, because the vulnerability hazard should not only consider the vulnerability itself attributes, such as vulnerability severity, but also need to consider the vulnerability external attributes, such as how much vulnerability affects assets. These issues also need to be studied.

6. Conclusion

This paper proposes an automatic vulnerability severity assessment method based on the distillation model to solve the problem of slow manual vulnerability assessment of CPS. The method first enhances the amount of text information using data augmentation and integration of optimal subsets to solve the data imbalance problem, then characterizes the vulnerability description text using the fine-tuned DistilBERT model to effectively extract the features of the text. Finally, the linear layer is used to classify the characterized vectors. In this paper, multiple groups of ablation experiments are conducted on data of 105,984 vulnerabilities, and the experimental results show that the current method achieves state-of-the-art performance in vulnerability severity assessment, and the method can achieve 97% assessment accuracy. In the future, we will consider CPS vulnerability assessment from different dimensions of vulnerability attributes, such as asset dimension and environment dimension, to build a set of metrics systems in line with vulnerability severity assessment and solve the problem of inaccurate vulnerability hazard assessment.

Data Availability

The dataset used in this paper is available from the corresponding author upon request. Researchers can also download from the NVD official website. Download links are in the 20th reference.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research was funded by the National Key Research and Development Program of China under grant number 2021YFB3100500.

References

[1] nvd.nist.gov, "CVSS severity distribution over time," 2022, https://nvd.nist.gov/general/visualizations/vulnerability-visualizations/cvss-severity-distribution-over-time.

[2] nvd.nist.gov, "National vulnerability database," 2022, https://nvd.nist.gov/general/nvd-dashboard.

[3] H. Chen, J. Liu, R. Liu, N. Park, and V. V. E. S. T. Subrahmanian, "A system for vulnerability exploit scoring & timing," in *Proceedings of the International Joint Conferences on Artifical Intelligence*, pp. 6503–6505, Macau, China, Auguest 2019.

[4] J. Ruohonen, "A look at the time delays in CVSS vulnerability scoring," *Applied Computing and Informatics*, vol. 15, pp. 129–135, 2019.

[5] S. Qaiser and R. Ali, "Text mining: use of TF-IDF to examine the relevance of words to documents," *International Journal of Computer Application*, vol. 181, pp. 25–29, 2018.

[6] J. B. Awotunde, C. Chakraborty, and A. E. Adeniyi, "Intrusion detection in industrial internet of things network-based on deep learning model with rule-based feature selection," *Wireless Communications and Mobile Computing*, vol. 2021, Article ID 7154587, 17 pages, 2021.

[7] B. Wang, A. Wang, F. Chen, Y. Wang, and C. C. J. Kuo, "Evaluating word embedding models: methods and experimental results," *APSIPA transactions on signal and information processing*, vol. 8, 2019.

[8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: pre-training of deep bidirectional transformers for language understanding," 2018, https://arxiv.org/abs/1810.04805.

[9] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: LSTM cells and network architectures," *Neural Computation*, vol. 31, pp. 1235–1270, 2019.

[10] H. C. Shin, H. R. Roth, M. Gao et al., "Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning," *IEEE Transactions on Medical Imaging*, vol. 35, pp. 1285–1298, 2016.

[11] first org, "Common vulnerability scoring system v3.1: specification document," 2022, https://www.first.org/cvss/v3.1/specification-document.

[12] P. Mell, K. Scarfone, and S. Romanosky, "A complete guide to the common vulnerability scoring system version 2.0," in *Proceedings of the Published by FIRST-Forum of Incident Response and Security Teams*, Seville, Spain, June 2007.

[13] P. Mell and K. Scarfone, "Improving the common vulnerability scoring system," *IET Information Security*, vol. 1, pp. 119–127, 2007.

[14] M. Schiffman, A. Wright, D. Ahmad, and G. Eschelbeck, *The Common Vulnerability Scoring System*, National Infrastructure Advisory Council, Vulnerability Disclosure Working Group, Washington, DC, USA, 2004.

[15] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: a survey," *International Journal of Computer Vision*, vol. 129, pp. 1789–1819, 2021.

[16] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," 2019, https://arxiv.org/abs/1910.01108.

[17] J. Yin, M. Tang, J. Cao, and H. Wang, "Apply transfer learning to cybersecurity: predicting exploitability of vulnerabilities by description," *Knowledge-Based Systems*, vol. 210, Article ID 106529, 2020.

[18] J. Wei and K. E. Zou, "Easy data augmentation techniques for boosting performance on text classification tasks," 2019, https://arxiv.org/abs/1901.11196.

[19] P. Somol, J. Novovicová, and P. Pudil, "Efficient feature subset selection and subset size optimization," *Pattern recognition recent advances*, vol. 1, 2010.

[20] nvd.nist.gov, "NVD data feeds," 2022, https://nvd.nist.gov/vuln/data-feeds.

[21] huggingface, "Distilbert-base-uncased," 2022, https://huggingface.co/distilbert-base-uncased.

[22] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "Xlnet: generalized autoregressive pretraining for language understanding," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[23] Y. Liu, M. Ott, N. Goyal et al., "A robustly optimized bert pretraining approach," 2019, https://arxiv.org/abs/1907.11692.

[24] F. Bieder, R. Sandkühler, and P. C. Cattin, "Comparison of methods generalizing max-and average-pooling," 2021, https://arxiv.org/abs/2103.01746.

[25] A. Khazaei, M. Ghasemzadeh, and V. Derhami, "An automatic method for CVSS score prediction using vulnerabilities description," *Journal of Intelligent and Fuzzy Systems*, vol. 30, pp. 89–96, 2016.

[26] G. Spanos, L. Angelis, and D. Toloudis, "Assessment of vulnerability severity using text mining," in *Proceedings of the 21st Pan-Hellenic Conference on Informatics*, pp. 1–6, Larissa, Greece, September 2017.

[27] S. Nakagawa, T. Nagai, H. Kanehara et al., "Character-level convolutional neural network for predicting severity of software vulnerability from vulnerability description," *IEICE - Transactions on Info and Systems*, vol. 102, pp. 1679–1682, 2019.

[28] P. Wang, Y. Zhou, B. Sun, and W. Zhang, "Intelligent prediction of vulnerability severity level based on text mining and XGBboost," in *Proceedings of the 2019 Eleventh International Conference on Advanced Computational Intelligence (ICACI)*, pp. 72–77, Guilin, China, June 2019.

[29] Z. Han, X. Li, Z. Xing, H. Liu, and Z. Feng, "Learning to predict severity of software vulnerability using only vulnerability description," in *Proceedings of the 2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 125–136, Shanghai, China, September 2017.

[30] K. Liu, Y. Zhou, Q. Wang, and X. Zhu, "Vulnerability severity prediction with deep neural network," in *Proceedings of the 2019 5th International Conference on Big Data and Information Analytics (BigDIA)*, pp. 114–119, Kunming, China, July 2019.

[31] G. Spanos, A. Sioziou, and L. Angelis, "WIVSS: a new methodology for scoring information systems vulnerabilities," in *Proceedings of the Proceedings of the 17th Panhellenic Conference on Informatics*, pp. 83–90, New York, NY, USA, September 2013.

[32] J. Luo, K. Lo, and H. Qu, "A software vulnerability rating approach based on the vulnerability database," *Journal of Applied Mathematics*, vol. 2014, Article ID 932397, 9 pages, 2014.

[33] W. Wang, F. Shi, M. Zhang, C. Xu, and J. Zheng, "A vulnerability risk assessment method based on heterogeneous

information network," *IEEE Access*, vol. 8, pp. 148315–148330, 2020.

[34] M. Ali, "Character level convolutional neural network for Arabic dialect identification," in *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2018)*, pp. 122–127, Santa Fe, NM, USA, August 2018.

[35] K. Ameri, M. Hempel, H. Sharif, J. Lopez, and K. C. Perumalla, "Cybersecurity claim classification by fine-tuning the BERT language model," *Journal of Cybersecurity and Privacy*, vol. 1, pp. 615–637, 2021.

[36] P. K. Kudjo, J. Chen, S. Mensah, R. Amankwah, and C. Kudjo, "The effect of Bellwether analysis on software vulnerability severity prediction models," *Software Quality Journal*, vol. 28, pp. 1413–1446, 2020.

[37] M. R. Shahid and H. Debar, "Cvss-Bert: Explainable natural language processing to determine the severity of a computer security vulnerability from its description," in *Proceedings of the 2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 1600–1607, Pasadena, CA, USA, December 2021.

[38] X. Gong, Z. Xing, X. Li, Z. Feng, and Z. Han, "Joint prediction of multiple vulnerability characteristics through multi-task learning," in *Proceedings of the 2019 24th International Conference on Engineering of Complex Computer Systems (ICECCS)*, pp. 31–40, Guangzhou, China, November 2019.

[39] J. C. Costa, T. Roxo, J. B. Sequeiros, H. Proença, and P. R. Inácio, *Predicting CVSS Metric via Description Interpretation*, IEEE Access, Piscataway, NJ, USA, 2022.

Hindawi
Security and Communication Networks
Volume 2022, Article ID 1181398, 15 pages
https://doi.org/10.1155/2022/1181398

Research Article

Mixed Strategy Analysis in Attack-Defense Game Model Based on 5G Heterogeneous Network of CPS Using ncPSO

Ning Liu, Qing-Wei Chai, Shangkun Liu, Fanying Meng, and Wei-Min Zheng

College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao, Shandong 266590, China

Correspondence should be addressed to Wei-Min Zheng; zhengwm901@126.com

Received 16 July 2022; Revised 20 September 2022; Accepted 5 October 2022; Published 7 November 2022

Academic Editor: Yuanyuan Huang

The development of the 5th Generation Mobile Communication Technology not only brings convenience to people but also brings many network security problems. Based on the static game theory of complete information, a game model of attack and defense with limited resources in heterogeneous networks of Cyber Physical Systems is established. This model analyzes the basic rules of the offensive and defensive strategies of both parties when the offensive and defensive resources are limited in the 5th Generation Mobile Communication Technology network environment. The model can also describe the interaction between attackers and defenders. A novel compact particle swarm optimization algorithm is proposed to solve the difficult problem of solving the Nash equilibrium of this game model. The simulation experiment proves that novel compact particle swarm optimization algorithm has good optimization ability and shows that the algorithm can effectively solve the Nash equilibrium of the model. The simulation experiment provides a strategic reference for the attack-defense game with limited resources.

1. Introduction

In recent years, the 5th Generation Mobile Communication Technology (5G) has developed rapidly and has been widely used in many aspects of the lives of people [1]. 5G is characterized by fast transmission rate and high stability [2]. In the future, 5G will be applied in cyber physical systems (CPS), industry, education, environment, medical care, transportation, and many other aspects to achieve real full coverage [3–5]. Based on these characteristics of 5G, heterogeneous networks of CPS are also developing rapidly. The core goal of heterogeneous networks is to achieve free connectivity of devices [6]. The Internet can be composed of multiple interconnected heterogeneous networks. The devices used to connect heterogeneous networks are routers. A heterogeneous network means that two or more wireless communication systems use different access technologies or use the same wireless access technology but belong to different wireless operators [7]. Heterogeneous networks can make use of existing multiple wireless communication systems, and through the integration of systems to make the multiple systems can learn from each other to meet the needs of future mobile communication services. The intelligent access means of multimode terminals is used by heterogeneous network to make a variety of different types of networks that can jointly provide wireless access anytime and anywhere for users [8]. The CPS is closely related to heterogeneous networks. The CPS is mainly divided into three layers: the perception layer, the network layer, and the control layer. The transmission of information mainly depends on the network layer. The connection of different devices among CPS can be regarded as a heterogeneous network [9]. The significance of CPS is to connect physical devices to the Internet, which enables physical devices to have five functions: computing, communication, precise control, remote coordination, and autonomy [10]. The communication is put on the same level as computing in CPS. The network scale of CPS surpasses that of existing industrial control network. The CPS is thought to connect the whole world. The development of 5G heterogeneous networks also brings opportunities for hackers [11–13]. Hackers can use advanced persistent threat (APT), denial of

service (DOS), and other attack methods to attack network devices of CPS and use eavesdropping, interference, and other attack methods to attack the information transmission links of CPS [14–16]. The attacks from hackers are characterized by diversity, large scale, and high density [17–19]. It brings great challenges for defenders to defend against attacks [20, 21]. Firewall and intrusion detection system (IDS) has a delay in attacks from hackers, and reasonable defense measures cannot be taken before attacks from hackers [22, 23]. Therefore, it is particularly important to carry out macro analysis of attack and defense behaviors of hacker and defender through game theory. Through this analysis, defenders can obtain the possible attack strategies of hackers before being attacked and then defenders can take effective defense strategies to reduce the loss.

There have been many studies using game theory to analyze the behaviors of hackers and defenders [24–26]. Researchers build different models for different application environments and use game theory for macro analysis [27–30]. Li et al. propose a zero-sum complete information static game model under the network topology graph and proposes a sensitive cost parameter for attacking and defensing [31]. He only considers two attack strategies, and the model is quite different from the actual situation, so this model is not universally applicable. Min builds a Colonel Blotto game using APT attack and defense [14]. And the Nash equilibrium of the model is solved. Simulation experiments show that a "hotbooting" CPU allocation method proposed by Min has lower computational complexity. Jiang proposes a network security defense graph by the state of devices [32]. Taking into account the three attributes of information confidentiality, availability, and integrity into the model, the optimal defense strategy is obtained through game theory. Leng et al. propose a three-player complete information static game model based on white hat, black hat, and software manufacturers. The simulation experiment shows that the model provides an effective strategy for software manufacturers to reduce the risk of software vulnerabilities [33]. Attiah et al. transform the three-strategy attack-defense game model into a two-strategy game model. And the mixed strategy of the model is solved [34]. Wang and Zeng propose a two-stage game in the process of attack and defense. In the first stage, the attacker scans the port of the defender, and the defender deceives or truthfully responds to the attacker. In the second stage, the two sides play a strategic game [35]. Ferguson-Walter et al. establish an incomplete information dynamic game model for network deception attacks to analyze the strategic choices of both attackers and defenders. This model is more in line with the current offensive and defensive scenarios and has strong reference significance [36]. Shen et al. propose a signal game model for the selection of intrusion detection strategies in the wireless sensor network environment. The model uses detection rate and false alarm rate and analyzes the impact of these two parameters on strategies selection [37].

Although there are many existing studies on attack-defense equilibrium based on game theory, there are not many studies on the attack and defense game in the 5G heterogeneous network environment of CPS. The biggest flaw of the existing research models is that they do not consider whether the communication links are attacked, but only consider whether the heterogeneous devices are attacked. Another flaw is that the scale of attack and defense is too small to meet the conditions of 5G. Intelligent computing provides a solution for solving Nash equilibrium of large-scale offensive and defensive games. The most used and most classic intelligent computing algorithm is the particle swarm optimization (PSO). The PSO is proposed by Kennedy and Russell in 1995 according to the foraging behavior of birds [38]. Different algorithms need to be optimized for different problems, so different heuristic algorithms are constantly proposed, such as genetic algorithm (GA) [39], whale optimization algorithm (WOA) [40], black hole (BH) [41], bat algorithm (BA) [42], and sine cosine algorithm (SCA) [43]. A novel compact particle swarm optimization (ncPSO) that uses less memory space and has stronger optimization ability is proposed to solve the Nash equilibrium of the game model proposed in this paper.

The main contributions of this paper are as follows:

(1) Establish a network topology graph based on 5G heterogeneous network of CPS to analyze the attack and defense behaviors. Increase the number of offensive and defensive strategies.

(2) Consider not only network devices being attacked but also communication links between devices being attacked.

(3) A ncPSO is proposed to solve the mixed strategy Nash equilibrium of the game model.

(4) The strategy selection problem of attackers and defenders under different attack resources and different defense resources is analyzed.

The rest of this paper is structured as follows. A 5G heterogeneous network topology diagram of CPS is constructed in Section 2. A attack and defense game model is established in Section 3. The cPSO is proposed and a method for solving mixed strategy Nash equilibrium using the cPSO is described in Section 4. Simulation experiments to discuss the mixed strategies of attackers and defenders under different resources are conducted in Section 5. A conclusion is given in Section 6.

2. System Model and Symbols

A heterogeneous network of CPS based on 5G can be described as an undirected connection graph. We use $G = (V, E)$ to represent the graph in this paper. $V = \{v_1, v_2, \ldots, v_{N_V}\}$ is a set containing all vertices. $N_V = |V|$ represents the total number of the vertices. Each v_i is a vertex in G and indicates a device in heterogeneous network of CPS. Devices in a heterogeneous network can be mobile phones, servers, smart appliances, routers, etc. $E = \{e_1, e_2, \ldots, e_{N_E}\}$ is a set containing all edges. $N_E = |E|$ represents the total number of the edges. Each e_i is an edge in G and indicates a type of communication link in heterogeneous network of CPS. Communication links in a heterogeneous network of CPS contain telephone

communication, mobile communication, point to point (P2P), etc. The number of all elements in the graph G is $N_G = N_V + N_E$. We can denote $G = \{g_1, g_2, \ldots, g_{N_G}\}$ by $G = (V, E)$. Every g_i can be an attack target for attacker and a protective target for defender.

We use a diagonal matrix $C_{N_v \times N_v}$ to describe the connectivity of graph G. The detailed representation of C is shown in (1). Let $G_{\max}$ denote the maximum connectivity of graph G.

$$C_{ij} = C_{ji} = \begin{cases} 1, & \text{vertex} i \text{ and vertex } j \text{ are connected,} \\ 0, & \text{vertex} i \text{and vertex } j \text{ are not connected.} \end{cases} \tag{1}$$

Since different devices and different communication links have different functions and roles in heterogeneous networks, they have different importance in networks. We use I_i to denote the importance of the ith element in graph G. For vertices, their importance is represented by the number of edges connected by them. For edges, their importance is represented by the smaller maximum connectivity of the subgraphs on either side of the edge. The importance of each element in graph G is shown in

$$I_i = \begin{cases} \sum_{j=1}^{N_V} C_{ij}, & \text{if } g_i \text{is} a \text{vertex,} \\ \min\left(\max\left(G_{1_{\max}}, G_{2_{\max}}\right)\right), & \text{if } g_i \text{isanedge,} \end{cases} \tag{2}$$

where G_1 and G_2 are subgraphs on either side of the edge g_i. $G_{1_{\max}}$, and $G_{2_{\max}}$ are the maximum connectivity of the subgraph G_1 and subgraph G_2.

Usually, the available resources for attackers and defenders to attack and defense network components are limited. Source_A and Source_D are used to denote the total available resources of attackers and defenders, respectively. An attacker can attack one or more elements according to the attack resources. For example, an attacker can use DOS to attack a server, use routing spoofing to attack a router, use flooding to attack a switch, and use APT to attack a specific device. They also can use eavesdropping, truncation, jamming, and tampering to attack the communication links between devices. The defender can defense the network elements using limited resources at the same time. For example, a defender can use blacklist to prohibit attackers attacking server, use flow cleaning to protect routers, and use protection software to defense application devices. Defenders also can use encryption, filter, and VPN to defense the communication links.

3. Attack and Defense Game Model

Based on the model graph proposed in Section 2, we establish a game model $M = (A, D, S_A, S_D, U_A, U_D)$ to analyze behaviors of attacker and defender. In this game model, the attacker and defender take measures at the same time and there is no cooperative relationship between them, but the benefit calculation is different for both attacker and defender, so this game model is a nonzero sum game model. The meanings of the six tuple in the game model M are as follows:

(1) A: An attacker in a 5G heterogeneous network. The attacker can adopt different attack strategies to attack different components in the heterogeneous network.

(2) D: A defender in a 5G heterogeneous network. The defender also can take feasible measures to protect components in the heterogeneous network.

(3) S_A: $S_A = \{S_{A_1}, S_{A_2}, \ldots S_{A_i}, \ldots, S_{A_M}\}$ is the attack action set for attacker, where M represents the total number of the attack strategies. $S_{A_i} = (a_1, a_2, \ldots, a_j, \ldots, a_{N_G}) \in S_A$ is an attack strategy combination in S_A. The value of a_j is shown in

$$a_j = \begin{cases} 1, & \text{if } g_j \text{is attacked,} \\ 0, & \text{if } g_j \text{is not attacked.} \end{cases} \tag{3}$$

(4) S_D: $S_D = \{S_{D_1}, S_{D_2}, \ldots S_{D_i}, \ldots, S_{D_N}\}$ is the defense action set for defender, where N represents the total number of the defense strategies. $S_{D_i} = (d_1, d_2, \ldots, d_j, \ldots, d_{N_G}) \in S_D$ is a defense strategy combination in S_D. The value of d_j is shown in

$$d_j = \begin{cases} 1, & \text{if } g_j \text{is defensed,} \\ 0, & \text{if } g_j \text{is not defensed.} \end{cases} \tag{4}$$

(5) U_A: $U_A(S_{A_i}, S_{D_j})$ is the attack profit after the attacker and defender adopt strategy S_{A_i} and S_{D_j}.

(6) U_D: $U_D(S_{A_i}, S_{D_j})$ is the defense profit after the attacker and defender adopt strategy S_{A_i} and S_{D_j}.

In the game, the available resources of both sides cannot be infinite. Taking human, material, and financial resources into consideration, Source_A and Source_D are used to denote the total available resources of attackers and defenders, respectively. Each element g_i in G has different attack costs and defense costs for attackers and defenders according to its own characteristics, denoted by Cost_i^A and Cost_i^D. Therefore, the offensive and defensive behaviors must meet the following:

$$\sum_{i=1}^{N_G} a_i \times \text{Cost}_i^A \le \text{Source}_A, \tag{5}$$

$$\sum_{i=1}^{N_G} d_i \times \text{Cost}_i^D \le \text{Source}_D. \tag{6}$$

Whether a network element g_i in G is breached depends on the strategies from attacker and defender. It can be described by

$$B_i = \begin{cases} 1, & (a_i = 1 \&\& d_i = 0), \\ 0, & (a_i = 1 \&\& d_i = 1) \| (a_i = 0), \end{cases} \tag{7}$$

where $B_i = 1$ represents that g_i is breached and $B_i = 0$ represents that g_i is not breached. When element g_i is

breached, we remove it from G, then we obtain a new graph G' after the game.

For the attacker, the gain function in the model M is represented by the loss connectivity of the graph G. For the defender, the gain function is represented by the maximum connectivity of the network after the game. We have assumed the maximum connectivity before game is denoted as $G_{\max}$ in Section 2. Then we assume the maximum connectivity after the game as $G_{\max'}$. So the U_A and U_D can be calculated by (8). The profit matrix can be represented in Table 1.

$$\begin{aligned} U_A &= G_{\max} - G_{\max'}, \\ U_D &= G_{\max'}, \\ \text{s.t.} \sum_{i=1}^{N_G} a_i \times \text{Cost}_i^A &\le \text{Source}_A, \\ \sum_{i=1}^{N_G} d_i \times \text{Cost}_i^D &\le \text{Source}_D. \end{aligned} \tag{8}$$

In the game, both the attacker and the defender hope to achieve their maximum benefits. So they choose to use the strategy that could maximize their benefits. They also need to consider the strategies form another on their own returns. Eventually, they will reach a strategic equilibrium. In the game, there may exist a pure strategy and a mixed strategy Nash equilibrium. We assume that the optimal policy of attacker is S_A^*, and the optimal policy of defender is S_D^*. If S_A^* and S_D^* satisfy (9), they are a set of Nash equilibrium solutions.

$$\begin{cases} \forall S_{A_i}, U_A\left(S_A^*, S_D^*\right) \ge U_A\left(S_{A_i}, S_D^*\right), \\ \forall S_{D_j}, U_D\left(S_A^*, S_D^*\right) \ge U_D\left(S_A^*, S_{D_j}\right). \end{cases} \tag{9}$$

4. Game Solution

A solution to solve the equilibrium strategy of M is proposed in this section. This paper takes the attacker as an example to discuss, and the analysis method of the defender is similar to attacker. In this model game M, there may be a pure strategy and a mixed-strategy Nash equilibrium.

For pure-policy Nash equilibrium, we use the traditional min-max theorem to solve [44]. The idea of this theorem is to choose the optimal strategy under various worst-case scenarios [45]. When the profit matrix of model M satisfies (10), strategy (S_A^*, S_D^*) is a pure strategy equilibrium.

$$\max_{1\le i\le M} \min_{1\le j\le N} U_{A_{ij}} = \min_{1\le j\le N} \max_{1\le i\le M} U_{A_{ij}} = U_{A_{i^*j^*}}. \tag{10}$$

Pure strategy is a special form of mixed strategy. When the probability of one strategy is 1, and the probabilities of other strategies are 0, it is a pure strategy. First of all, a strategy choice is given for one participant, another participant chooses the strategy that brings the highest profit among the different strategies. Second, the optimal strategy of another participant is given, and the first participant compare whether the strategy which another participant chooses is also optimal for him. Third, if the strategy combination is the optimal strategy for both participants, then it is a pure strategy Nash equilibrium of the game. When a Nash equilibrium is reached, both offense and defense are reluctant to change their strategies because changing strategies will make their gains less. When there is more than one pure-strategy Nash equilibrium, both sides can choose mixed strategies to maximize their gains. Mixed strategies also occur when the game does not have a pure strategy Nash equilibrium. Taking the chicken game as an example, there are two pure strategy Nash equilibrium in the chicken game. But to participants, they want to take two strategies with a certain probability combination to obtain greater benefits, so the Nash equilibrium of mixed strategies is more important. Therefore, we mainly solve the Nash equilibrium of the mixed strategy. Mixed strategy means that players choose strategies according to probability. We set $P = (p_1, p_2, \ldots, p_M)$ and $Q = (q_1, q_2, \ldots, q_N)$ to represent the probability of the attacker and defender using different strategies, respectively. Therefore, the P and Q should satisfy

$$\left\{ \sum_{i=1}^{M} p_i = 1, \sum_{j=1}^{N} q_j = 1. \right. \tag{11}$$

According to Table 1, the expected returns of the attacker U_A' and the defender U_D' can be expressed as (12). Taking the attacker as an example, if P^* and Q^* are a Nash equilibrium of mixed strategies, then any individual modification of P or Q alone cannot increase the profit. Another solution to mixed strategies is that optimal mixed strategy of every player must make the expected profit of another choosing different strategies the same. The strategy of attacker should consider the profit of defender, it should satisfy (13). The strategy of defender should consider the profit of attacker, it should satisfy

$$\begin{aligned} U_A' &= \max\left(\sum_{i=1}^{M}\sum_{j=1}^{N} p_i \cdot q_j \cdot U_A\left(S_{A_i}, S_{D_j}\right)\right), \\ U_D' &= \max\left(\sum_{i=1}^{M}\sum_{j=1}^{N} p_i \cdot q_j \cdot U_D\left(S_{A_i}, S_{D_j}\right)\right), \\ s.t. \sum_{i=1}^{N_G} a_i \times \text{Cost}_i^A &\le \text{Source}_A, \\ \sum_{i=1}^{N_G} d_i \times \text{Cost}_i^D &\le \text{Source}_D, \\ \sum_{i=1}^{M} p_i &= 1, \\ \sum_{i=1}^{N} q_i &= 1. \end{aligned} \tag{12}$$

$$\begin{aligned} \sum_{i=1}^{M} p_i \cdot U_D\left(S_{A_i}, S_{D_1}\right) &= \sum_{i=1}^{M} p_i \cdot U_D\left(S_{A_i}, S_{D_2}\right) = \ldots \\ &= \sum_{i=1}^{M} p_i \cdot U_D\left(S_{A_i}, S_{D_N}\right), \end{aligned} \tag{13}$$

Table 1: The profit matrix of attacker and defender.

	S_{D_1}	S_{D_2}	$\cdots$	S_{D_N}
S_{A_1}	$U_{A_{11}}, U_{D_{11}}$	$U_{A_{12}}, U_{D_{12}}$	$\cdots$	$U_{A_{1N}}, U_{D_{1N}}$
S_{A_2}	$U_{A_{21}}, U_{D_{21}}$	$U_{A_{22}}, U_{D_{22}}$	$\cdots$	$U_{A_{2N}}, U_{D_{2N}}$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
S_{A_M}	$U_{A_{M1}}, U_{D_{M1}}$	$U_{A_{M2}}, U_{D_{M2}}$	$\cdots$	$U_{A_{MN}}, U_{D_{MN}}$

$$\sum_{j=1}^{N} p_j \cdot U_A\left(S_{A_1}, S_{D_j}\right) = \sum_{j=1}^{N} p_j \cdot U_A\left(S_{A_2}, S_{D_j}\right) = \cdots = \sum_{j=1}^{N} p_j \cdot U_A\left(S_{A_M}, S_{D_j}\right). \tag{14}$$

With the development of 5G, the means of network attack are diversified, large scale, and intelligent. The equilibrium solution under large-scale attack and defense strategies is a difficult problem. The high-speed and low-latency 5G make offensive and defensive strategies grow exponentially. Under the large-scale attack and defense strategies, the traditional Nash equilibrium solution method is no longer applicable. As an optimization algorithm, intelligent computing has a good effect on solving high-dimensional complex problems.

A ncPSO is proposed to solve the Nash equilibrium of mixed strategies. The ncPSO is an improvement over PSO. The position of each particle donated as X_i represents a strategic choice of the attacker. PSO will record the global optimal position gBest and the individual optimal position $pBest$ of the particle swarm. Then particles move according to gBest and pBest, and the movement equation refers to (15) and (16). w is the inertia weight, and g is the number of iterations.

$$V_i^{g+1} = wV_i^g + c_1 \times r\text{and} \times \left(p\text{Best}_i - X_i^g\right) + c_2 \times r\text{and} \times \left(g\text{Best} - X_i^g\right), \tag{15}$$

$$X_i^{g+1} = X_i^g + V_i^{g+1}. \tag{16}$$

The movement process of the particle swarm is shown in Figure 1. V_1, V_2, and V_3 represent three speeds in (15). V_i^{g+1} is the combination of the three speeds. The new position of X_i^{g+1} is based on X_i^g and V_i^{g+1}.

PSO is an iterative optimization algorithm, and gBest and pBest are updated according to the fitness value of particles. Considering that PSO requires larger memory, we apply compact strategy and propose ncPSO to optimize and reduce memory usage [46]. Instead of recording the position of each particle in each iteration, the compact strategy uses a mathematical distribution to record the position of the entire swarm of particles. The compact strategy that we proposed uses a Pareto distribution to describe the location of the particle swarm. The Pareto distribution contains three important parameters: shape parameter k, scale parameter σ, and threshold parameter θ [47]. Among the three parameters, AA and BB play a decisive role in the optimization and convergence of the algorithm, so AA and BB are used to update the particle swarm position. Taking 50 particles with 40 dimensions as an example, the original PSO needs to record the position of each particle in each dimension, so $50 \times 40 = 2000$ memory units are required. The ncPSO only needs to record the AA and BB of entire swarm of particles in each dimension, so the ncPSO only needs $2 \times 40 = 80$ memory units. Compared with traditional PSO, ncPSO can reduce a large number of memory units. The process of the compact strategy is shown as follows. First, generate a probability distribution function (PDF) according to θ and σ. Second, calculate cumulative distribution function (CDF) by PDF. Finally, a random number will be generated to calculate an inverse cumulative distribution function (iCDF). The value of iCDF is the position of the particle swarm. The figures of PDF and CDF of Pareto are shown in Figure 2. The equations of PDF and CDF of Pareto are shown in (17) and (18).

$$\text{PDF} = \frac{1}{\sigma}\left(1 + k\frac{(x-\theta)}{\sigma}\right)^{-1-1/k}, \tag{17}$$

$$\text{CDF} = \begin{cases} 1 - \left(1 + k\dfrac{(x-\theta)}{\sigma}\right)^{-1/k}, & k \neq 0, \\ 1 - \exp\left(-\dfrac{(x-\theta)}{\sigma}\right), & k = 0. \end{cases} \tag{18}$$

The ncPSO generates a winner and a loser through a competitive strategy and then updates σ and θ through the winner and loser. The update formulas for σ and θ are shown in (19) and (20), where Np is the virtual number of particles. For the attacker, the condition for winning the competition in the model can be expressed as how to choose the probability of using each attack strategy to make the defender get the same benefit adopting any defensive strategy. The ith dimension of the particle represents the probability that the attacker chooses the attack strategy S_{A_i}. The fitness value of ncPSO can be expressed as the standard deviation of the profit of defender. Its mathematical representation is shown in

$$\sigma^{g+1} = \sqrt{\left(\sigma^g\right)^2 + \left(\theta^g\right)^2 - \left(\theta^{g+1}\right)^2 + \frac{1}{Np}\left(\text{winner}^2 - \text{loser}^2\right)}, \tag{19}$$

$$\theta^{g+1} = \sigma^g + \frac{1}{Np}\left(\text{winner} - \text{loser}\right), \tag{20}$$

$$\text{Total Fitness}U_A(j) = \sum_{i=1}^{M} p_i \cdot U_D\left(S_{A_i}, S_{D_j}\right) j = 1, 2, \ldots, N\text{fitness}U_A = \text{std}\left(\text{Total Fitness}U_A\right). \tag{21}$$

For an attacker, let P^* be the optimal probability distribution for choosing different attack strategies. The process of the ncPSO to calculate the P^* is shown in Algorithm 1. In the same way, we can obtain the optimal probability distribution for the defender to choose different defense strategies.

The whole algorithm can be divided into the following six parts:

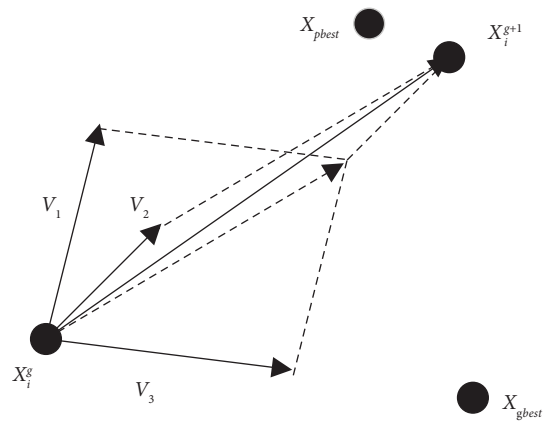


Figure 1: The movement process of the particle swarm.

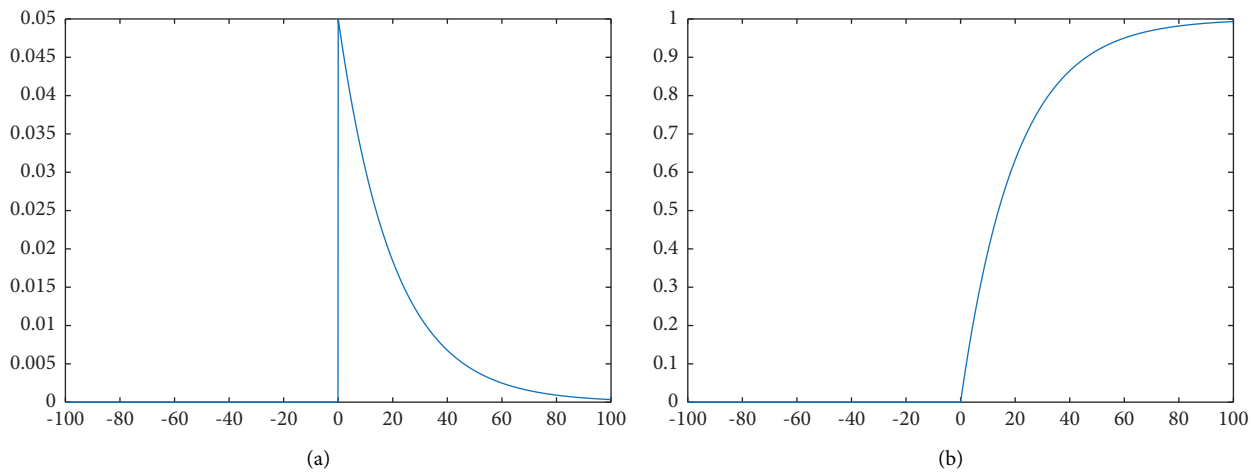


Figure 2: PDF and CDF of Pareto distribution. (a) PDF of Pareto distribution. (b) CDF of Pareto distribution.

(1) The network model is established according to the connection relationship between the actual heterogeneous network devices.

(2) Calculate the importance of each element.

(3) Initialize attack and defense costs for each element. Initialize the total available resources of the attacker and defender.

(4) Build a set of strategies for attacker and defender to attack and defense network element.

(5) Calculate the profit matrix for attacker and defender.

(6) Calculate the mixed strategy Nash equilibrium of attacker and defender using ncPSO.

5. Simulation Experiment and Analysis

In this section, the performance of ncPSO is compared with GA, PSO, WOA, BH, BA, and SCA in 28 test functions in CEC2013. CEC2013 contains 28 different performance test functions, which have good representation. Then, we research the mixed strategy Nash equilibrium using a simple topology diagram. But in real life, the topology diagram of heterogeneous network is much more complicated. Our purpose is to analyze the attack and defense behavior under different attack and defense resources, so using a simple topology diagram can also get the desired results. Four sets of simulation experiments are tested to illustrate the validity of the equilibrium solution and the feasibility of the assignment of offensive and defensive strategies in the game model. First, simulation experiments are carried out to demonstrate the effectiveness of solving Nash equilibrium using ncPSO. Second, simulation experiments are carried out to demonstrate the losses of the defender under different attack and defense resources. Third, strategies of attacker and defender with different defense resources are discussed through simulation experiments. Lastly, strategies of attacker and defender with different attack resources are discussed through simulation experiments. The simulation tools in this section all use Matlab2021B.

5.1. Performance Test Experiment of ncPSO and Other Algorithms. In this subsection, the performance of different algorithms is compared on 28 test functions of CEC2013. Different algorithms search for the minimum value on the test function. The smaller the function value found, the stronger the optimization ability of the algorithm. Each

```
Initialize the θ and σ of Pareto
Initialize the X by uniform distribution
Initialize the V by uniform distribution
Initialize the gBest, pBest, fitnessPBest, fitnessGBest
while g < = max_iteration do
pBest = compact (θ, σ) according to (17) and (18)
Calculate the fitnessPBest of the pBest by (21)
Update X and V of each particle according to (15) and (16)
Calculate the fitness of the new X by (21)
if fitness(X) < fitnessPBest then
winner = X
loser = pBest
else
winner = pBest
loser = X
end if
Update θ and σ according to (19) and (20)
pBest = winner
fitnessPBest = fitness(winner)
if fitness(X) < fitnessGBest then
gBest = X
fitnessGBest = fitness(X)
end if
g = g + 1
end while
```

Algorithm 1: The process for calculating P^* using cPSO.

algorithm is run 20 times on each test function, and the mean of the 20 experimental data is taken as the final experimental result. In addition to the performance test experiments, this section also performs Wilcoxon signed rank sum test with significant level $\alpha = 0.05$ on the test results to illustrate the validity of the experimental data. According to the relevant references of different algorithms, the parameter settings of different algorithms are shown in Table 2.

In Table 2, D represents the dimension of the problem, N represents the number of populations, and other parameters are unique to each algorithm and take values according to relevant references. The results of performance test experiments and rank sum verification are shown in Table 3. In Table 3, $>, =, <$, respectively, indicates that ncPSO has better, same, and worse optimization effects on the test function than other algorithms.

The last row of Table 3 gives the number of better, the same, and worse performance of ncPSO compared to other algorithms in the 28 test functions. The experimental results in Table 3 show that ncPSO has the same optimization effect as all optimization algorithms on the two test functions $f8$ and $f20$. Compared with GA, ncPSO performs better than GA in other 26 test functions. Compared with PSO, ncPSO performs better than PSO on 18 test functions, but not as good as PSO on 8 test functions. Compared with WOA and SCA, ncPSO performs better than WOA and SCA on 24 test functions, but worse than WOA and SCA on $f4$ and $f25$. Compared with BH, ncPSO performs better than BH on 21 test functions, but not as good as BH on 4 test functions. Compared with BA, ncPSO performs better than BA on 18 test functions, but worse than BA on 7 test functions.

Table 2: Parameter settings for different algorithms.

Name	Parameter
ncPSO	$D = 50$; $c1 = 2.0$; $c2 = 2.0$;
GA	$D = 50$; $N = 60$; crossover rate = 0.01; mutation rate = 0.9
PSO	$D = 50$; $N = 60$; $c1 = 1.0$; $c2 = 2.0$
WOA	$D = 50$; $N = 60$; probability switch = 0.5
BH	$D = 50$; $N = 60$
BA	$D = 50$; $N = 60$; pulse rate = 0.5; loudness = 0.6; $f_{\min} = 0$; $f_{\max} = 1$
SCA	$D = 50$; $N = 60$; probability switch = 0.5

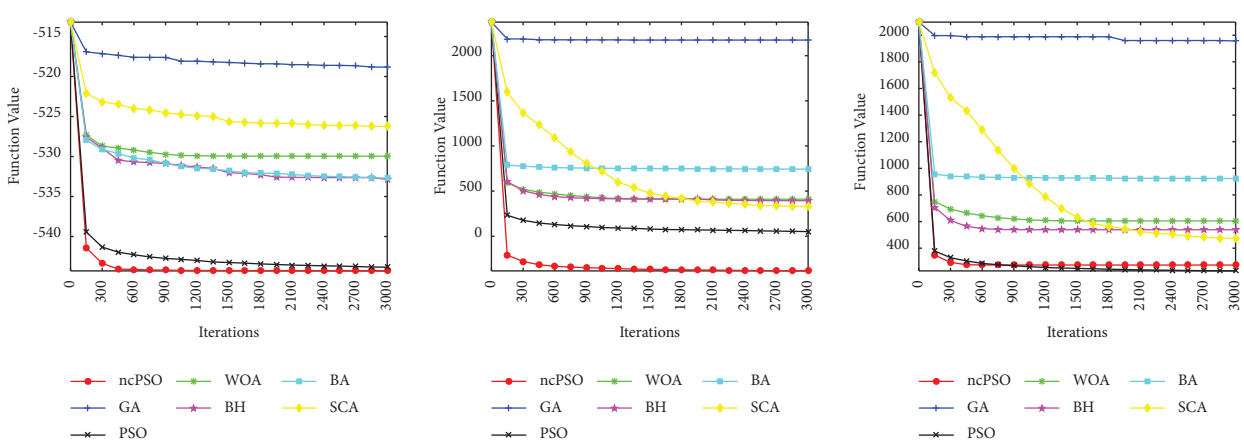
In order to better show the optimization performance of the algorithms, 9 optimization process diagrams with obvious effects are selected for display in Figure 3. After the algorithm iteration in each graph, the lower the curve, the stronger the optimization ability of the algorithm, and the faster the curve declines, the faster the convergence speed of the algorithm.

As can be seen in Figure 3, ncPSO has better optimization performance and faster convergence speed than other algorithms in functions $f9$, $f11$, $f13$, $f14$, $f17$, $f22$, $f24$, and $f26$. On the function $f12$, ncPSO has worse optimization performance compared with PSO, but has better optimization performance than GA, WOA, BH, BA, and SCA.

5.2. Effectiveness of Solving Nash Equilibrium Using ncPSO. The graph that we use in the simulation experiment is shown in Figure 4. Because a heterogeneous network of CPS is formed by various devices and different access technologies, the heterogeneous network of CPS is simplified as Figure 4.

Table 3: Experiment results of performance test and rank sum test.

	GA		PSO		WOA		BH		BA		SCA		ncPSO
f1	1.63E + 05	>	−1.33E + 03	>	−1.34E + 03	>	−1.40E + 03	=	−1.39E + 03	>	2.91E + 04	>	−1.40E + 03
f2	5.40E + 09	>	8.83E + 06	>	8.07E + 07	>	2.78E + 07	>	5.86E + 06	>	5.49E + 08	>	1.74E + 06
f3	2.37E + 20	>	2.65E + 09	>	3.47E + 10	>	7.94E + 09	>	6.09E + 08	<	1.05E + 11	>	1.04E + 09
f4	6.79E + 05	>	1.11E + 03	<	5.90E + 04	<	3.18E + 04	<	1.91E + 04	<	6.65E + 04	<	8.12E + 04
f5	8.68E + 04	>	−9.76E + 02	>	−7.96E + 02	>	−9.01E + 02	>	−9.96E + 02	>	2.26E + 03	>	−1.00E + 03
f6	2.78E + 04	>	−8.04E + 02	>	−6.79E + 02	>	−7.96E + 02	>	−8.28E + 02	<	1.12E + 03	>	−8.23E + 02
f7	6.05E + 06	>	−6.76E + 02	>	9.12E + 02	>	−6.19E + 02	>	1.26E + 04	>	−5.81E + 02	>	−6.84E + 02
f8	−6.79E + 02	=	−6.79E + 02	=	−6.79E + 02	=	−6.79E + 02	=	−6.79E + 02	=	−6.79E + 02	=	−6.79E + 02
f9	−5.19E + 02	>	−5.44E + 02	>	−5.30E + 02	>	−5.31E + 02	>	−5.33E + 02	>	−5.26E + 02	>	−5.44E + 02
f10	2.32E + 04	>	−4.43E + 02	>	−1.79E + 02	>	−4.67E + 02	>	−4.96E + 02	>	3.38E + 03	>	−4.99E + 02
f11	2.17E + 03	>	5.13E + 01	>	4.10E + 02	>	4.33E + 02	>	7.43E + 02	>	3.31E + 02	>	−3.84E + 02
f12	1.96E + 03	>	2.29E + 02	<	6.05E + 02	>	5.53E + 02	>	9.23E + 02	>	4.75E + 02	>	2.73E + 02
f13	2.15E + 03	>	4.06E + 02	>	7.97E + 02	>	6.43E + 02	>	1.21E + 03	>	5.60E + 02	>	3.51E + 02
f14	1.66E + 04	>	6.47E + 03	>	9.10E + 03	>	8.57E + 03	>	8.88E + 03	>	1.35E + 04	>	1.84E + 03
f15	1.61E + 04	>	8.53E + 03	<	1.15E + 04	>	8.89E + 03	<	9.14E + 03	<	1.48E + 04	>	9.55E + 03
f16	2.05E + 02	>	2.03E + 02	>	2.03E + 02	>	2.02E + 02	>	2.02E + 02	>	2.04E + 02	>	2.02E + 02
f17	5.37E + 03	>	7.99E + 02	>	1.45E + 03	>	1.35E + 03	>	2.69E + 03	>	1.30E + 03	>	3.51E + 02
f18	5.51E + 03	>	8.68E + 02	<	1.56E + 03	>	1.45E + 03	>	2.81E + 03	>	1.41E + 03	>	9.85E + 02
f19	2.32E + 07	>	5.30E + 02	>	6.81E + 02	>	6.14E + 02	>	5.64E + 02	>	4.31E + 04	>	5.02E + 02
f20	6.25E + 02	=	6.24E + 02	=	6.25E + 02	=	6.24E + 02	=	6.25E + 02	=	6.24E + 02	=	6.25E + 02
f21	1.27E + 04	>	1.67E + 03	>	1.93E + 03	>	1.68E + 03	>	1.48E + 03	<	4.58E + 03	>	1.64E + 03
f22	1.87E + 04	>	1.02E + 04	>	1.29E + 04	>	1.27E + 04	>	1.24E + 04	>	1.54E + 04	>	4.31E + 03
f23	1.83E + 04	>	1.20E + 04	<	1.41E + 04	>	1.28E + 04	<	1.19E + 04	<	1.61E + 04	>	1.40E + 04
f24	1.99E + 03	>	1.38E + 03	>	1.41E + 03	>	1.43E + 03	>	1.44E + 03	>	1.43E + 03	>	1.36E + 03
f25	1.73E + 03	>	1.54E + 03	<	1.53E + 03	>	1.54E + 03	<	1.47E + 03	<	1.55E + 03	>	1.60E + 03
f26	1.79E + 03	>	1.65E + 03	>	1.67E + 03	>	1.61E + 03	>	1.68E + 03	>	1.62E + 03	>	1.48E + 03
f27	4.72E + 03	>	3.32E + 03	<	3.56E + 03	>	3.56E + 03	>	3.47E + 03	>	3.64E + 03	>	3.32E + 03
f28	1.70E + 04	>	4.44E + 03	<	8.75E + 03	>	7.50E + 03	>	1.04E + 04	>	6.62E + 03	>	5.16E + 03
>/ = /<	26/2/0		18/2/8		24/2/2		21/3/4		18/3/7		24/2/2		



(a) (b) (c)

Figure 3: Continued.

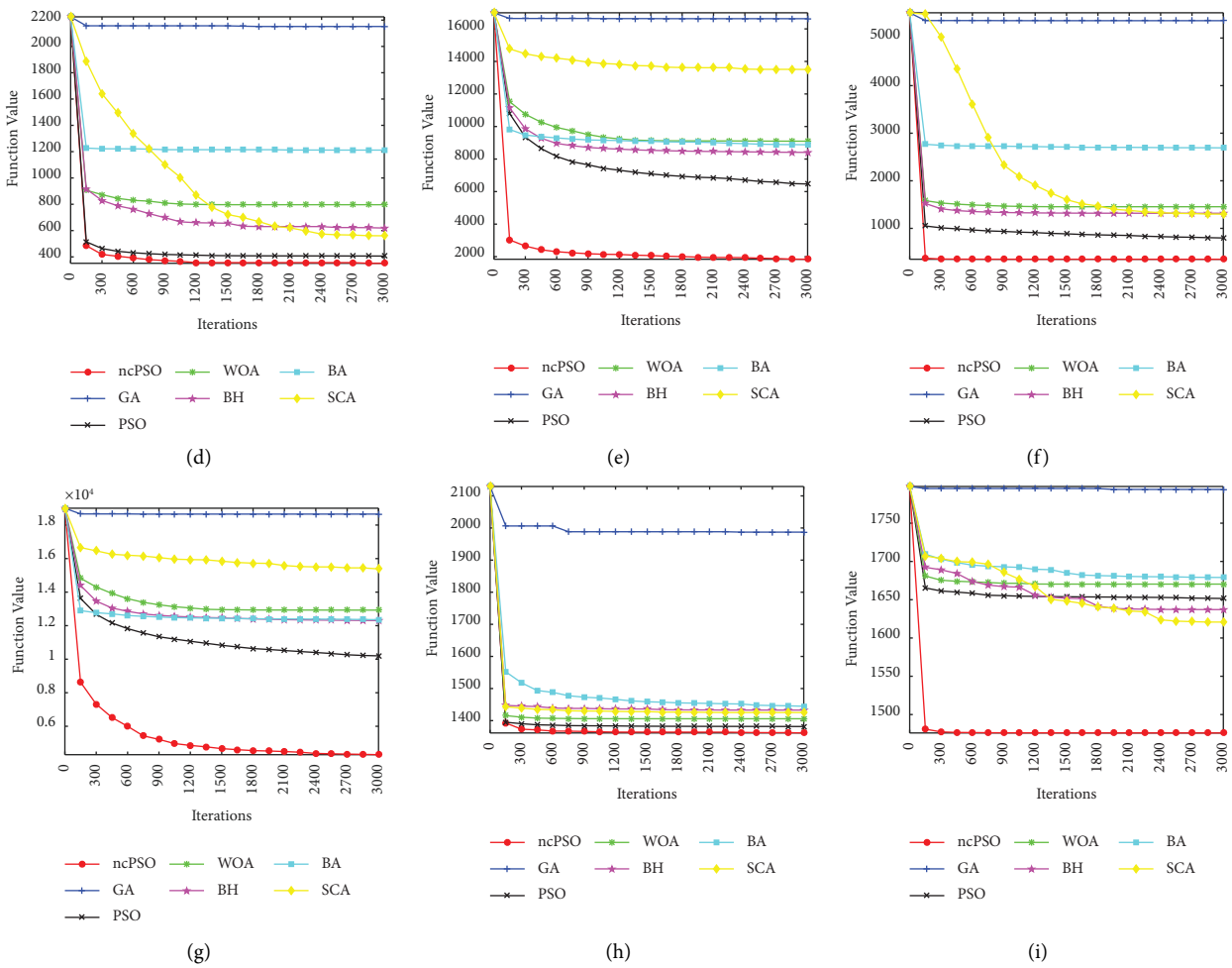


Figure 3: Performance comparison between GA, PSO, WOA, BH, BA, SCA, and ncPSO. (a) $f9$. (b) $f11$. (c) $f12$. (d) $f13$. (e) $f14$. (f) $f17$. (g) $f22$. (h) $f24$. (i) $f26$.

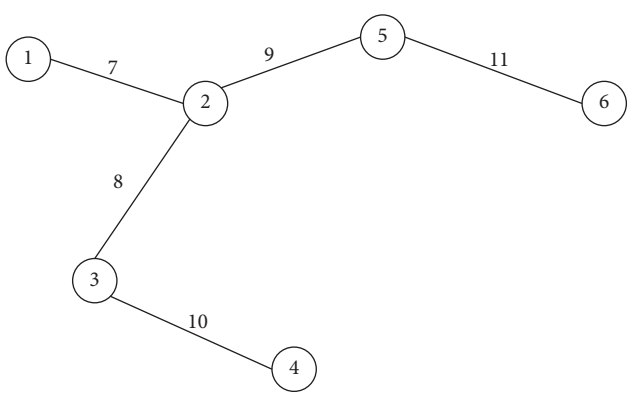


Figure 4: The graph of the simulation experiment.

Each node in the diagram represents a terminal device in the network of CPS, such as terminal servers, mobile phones, sensors, and drones. Each edge in the graph represents a communication link in the network of CPS, such as optical fibers to transmit signals, radiowaves to transmit signals. We set the cost of attacking and defending each element in the graph to be 1. When the offensive and defensive resources are 2, 4, 6, 8, and 10, respectively, the number of strategies of attackers and defenders is shown in Table 4.

From Table 4, it can be seen that there are a large number of offensive and defensive strategies compared to previous studies. There are so many strategies in the small network in Figure 4, which is in line with the large scale of attack and defense in the 5G heterogeneous network environment.

In this subsection, simulation experiments are performed to demonstrate the effectiveness of using intelligent computing to solve the Nash equilibrium. When the defense resources are 4 and 6, respectively, attack and defense experiments are performed ten times when the attack resources are 2, 4, 6, 8, and 10, respectively. A solution to mixed strategies is that optimal mixed strategy of every player must make the expected profit of another choosing different strategies the same. In other words, no single player can change the combination of strategies he adopts to increase his own profit.

First of all, the defense balance strategy is found through ncPSO. Then, ten times attacks are randomly performed under the defense equilibrium strategy calculated by ncPSO,

Table 4: The number of strategies under different offensive and defensive resources.

	2	4	6	8	10
The number of strategies	$C_{11}^{2} = 55$	$C_{11}^{4} = 330$	$C_{11}^{6} = 462$	$C_{11}^{8} = 165$	$C_{11}^{10} = 10$

Table 5: The attack profit when defense resource is 4 and attack resources are 2, 4, 6, 8, and 10, respectively.

	1	2	3	4	5	6	7	8	9	10	Average	Std
2	1.5939	1.5756	1.5776	1.6153	1.5623	1.5709	1.5761	1.5824	1.5859	1.5787	1.5819	**0.0145**
4	2.7161	2.7045	2.7013	2.6899	2.7041	2.6968	2.6870	2.6902	2.7026	2.7084	2.7001	**0.0091**
6	3.5247	3.5234	3.5240	3.5326	3.5276	3.5248	3.5292	3.5229	3.5309	3.5194	3.5259	**0.0040**
8	4.1195	4.1272	4.1271	4.1162	4.1216	4.1284	4.1225	4.1187	4.1272	4.1249	4.1233	**0.0043**
10	4.7563	4.7561	4.7562	4.7564	4.7562	4.7559	4.7564	4.7561	4.7561	4.7565	4.7562	**0.0002**

The bold values are the standard deviations of the ten experiments. The bolded data are used to show that ncPSO is effective in solving Nash equilibrium.

Table 6: The attack profit when defense resource is 6 and attack resources are 2, 4, 6, 8, and 10, respectively.

	1	2	3	4	5	6	7	8	9	10	Average	Std
2	1.1381	1.1597	1.1563	1.1622	1.1636	1.1473	1.1629	1.1638	1.1544	1.1611	1.1569	**0.0084**
4	2.1160	2.1169	2.1136	2.1206	2.1188	2.1157	2.1238	2.1208	2.1183	2.1199	2.1184	**0.0030**
6	2.8821	2.8819	2.8810	2.8752	2.8750	2.8779	2.8787	2.8811	2.8809	2.8796	2.8793	**0.0026**
8	3.5171	3.5166	3.5126	3.5154	3.5144	3.5183	3.5045	3.5107	3.5087	3.5192	3.5137	**0.0047**
10	4.0934	4.0932	4.0937	4.0931	4.0934	4.0939	4.0931	4.0937	4.0932	4.0930	4.0934	**0.0003**

The bold values are the standard deviations of the ten experiments. The bolded data are used to show that ncPSO is effective in solving Nash equilibrium.

and the profit of attacker is calculated. The experimental results are shown in Tables 5 and 6.

The first row in Tables 5 and 6 represents ten times random attacks, the average and standard deviation of these ten attacks. The first column in Tables 5 and 6 represents different attack resources. It can be seen from Tables 5 and 6 that under the condition of the same attack resources, the profit of the attacker is roughly the same in ten attacks. The values of the standard deviations of ten attacks are very small. The standard deviation indicates how spread out the data are. The smaller the standard deviation value, the smaller the difference between the data. The small standard deviation in the experimental results also indicates that the profit from ten attacks are approximately equal. This proves that using ncPSO to solve mixed-strategy Nash equilibrium is effective.

5.3. Losses of Defender under Different Attack and Defense Resources. In this subsection, simulation experiments are performed to discuss the losses of defender under different attack and defense resources. When the attack resources and defense resources are 2, 4, 6, 8, and 10, respectively, ten times simulation experiments are performed to calculate the losses of defender. Then the average and standard deviation of the losses of defender over ten experiments are calculated. The results of ten times experiments are shown in Table 7.

Table 5 shows the defense losses obtained by using ncPSO to solve the Nash equilibrium under different attack and defense resources. The smaller standard deviation of each group of experiments indicates the stability of the experiment. The experimental results show that the more defense resources, the less defense loss of the defender. The more resources to attack, the greater the defense loss of the defender. At the same time, the experimental results show that when the offensive and defensive resources are equal and equal to about half of all the offensive and defensive resources, it is most unfavorable to the defender.

5.4. Strategies of Attacker and Defender with Different Defense Resources. In this section, the attack resource is fixed as 4, and the strategy selection of the attacker and the defender is discussed when the defense resource is 2, 4, 6, 8, and 10, respectively. In order to prevent the chance of the results, each group of experiments was carried out 10 times, and then the mean value was taken as the experimental result. After using ncPSO to calculate the mixed-strategy Nash equilibrium of both players of the game, we map it to the probability of attacking and defending each element to analyze the attack and defense behaviors. The experimental results are shown in Figure 5.

From subfigures a, b, and c, it can be seen that when there are fewer defensive resources, the defender will spend more resources on elements of higher importance. The attacker will attack less important elements to avoid the defense of defender to obtain a higher profit. As can be seen from subfigures d and e, when defense resources are sufficient, the defender will evenly allocate defense resources to each element. Because the attacker does not know the resource allocation of the defender, he will guess that the defender allocates most of the resources to the elements of high importance, so the attacker will still allocate the attack resources to the elements of lower importance.

Table 7: The losses of defender under different attack and defense resources.

		Defense resources									
		2		4		6		8		10	
		Average	Std	Average	Std	Average	Std	Average	Std	Average	Std
Attack resources	2	−2.1073	0.0296	−1.5819	0.0145	−1.1569	0.0084	−0.7155	0.0021	−0.2417	0.0003
	4	−3.3408	0.0121	−2.7001	0.0091	−2.1184	0.0030	−1.3748	0.0011	−0.4854	0.0001
	6	−4.2067	0.0119	−3.5259	0.0040	−2.8793	0.0026	−1.9765	0.0011	−0.7312	0.0002
	8	−4.7108	0.0068	−4.1233	0.0043	−3.5137	0.0047	−2.4765	0.0017	−0.9691	0.0002
	10	−4.9809	0.0001	−4.7562	0.0002	−4.0934	0.0003	−3.0077	0.0018	−1.2180	0.0003

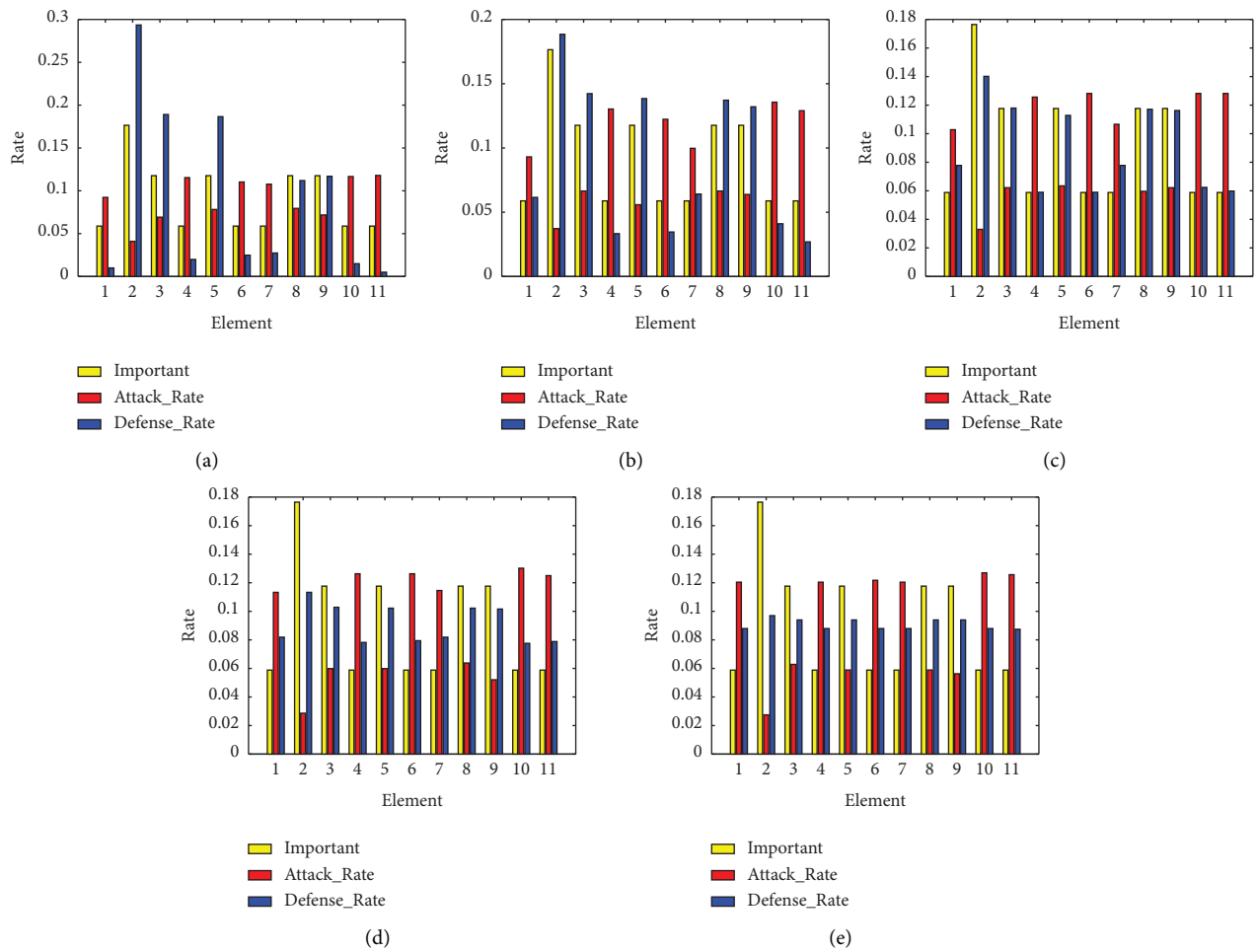


Figure 5: Experimental results under different defense resources. (a) Defense resources = 2. (b) Defense resources = 4. (c) Defense resources = 6. (d) Defense resources = 8. (e) Defense resources = 10.

5.5. Strategies of Attacker and Defender with Different Attack Resources. In this section, the defense resource is fixed as 4, and the strategy selection of the attacker and the defender is discussed when the attack resource is 2, 4, 6, 8, and 10, respectively. In order to prevent the chance of the results, each group of experiments was carried out 10 times, and then the mean value was taken as the experimental result. After using ncPSO to calculate the mixed-strategy Nash equilibrium of both players of the game, we map it to the probability of attacking and defending each element to analyze the attack and defense behaviors. The experimental results are shown in Figure 6.

As can be seen from subfigures a, b, and c, when there are few attack resources, the defender will allocate more defense resources to the elements with high importance, so that the attacker will attack the elements that allocate less defense resources to obtain profit successfully. As can be seen from subfigures d and e, when there are many attack resources, the attacker no longer considers the strategy choice of defender, but randomly attacks each element. In this case, the defender

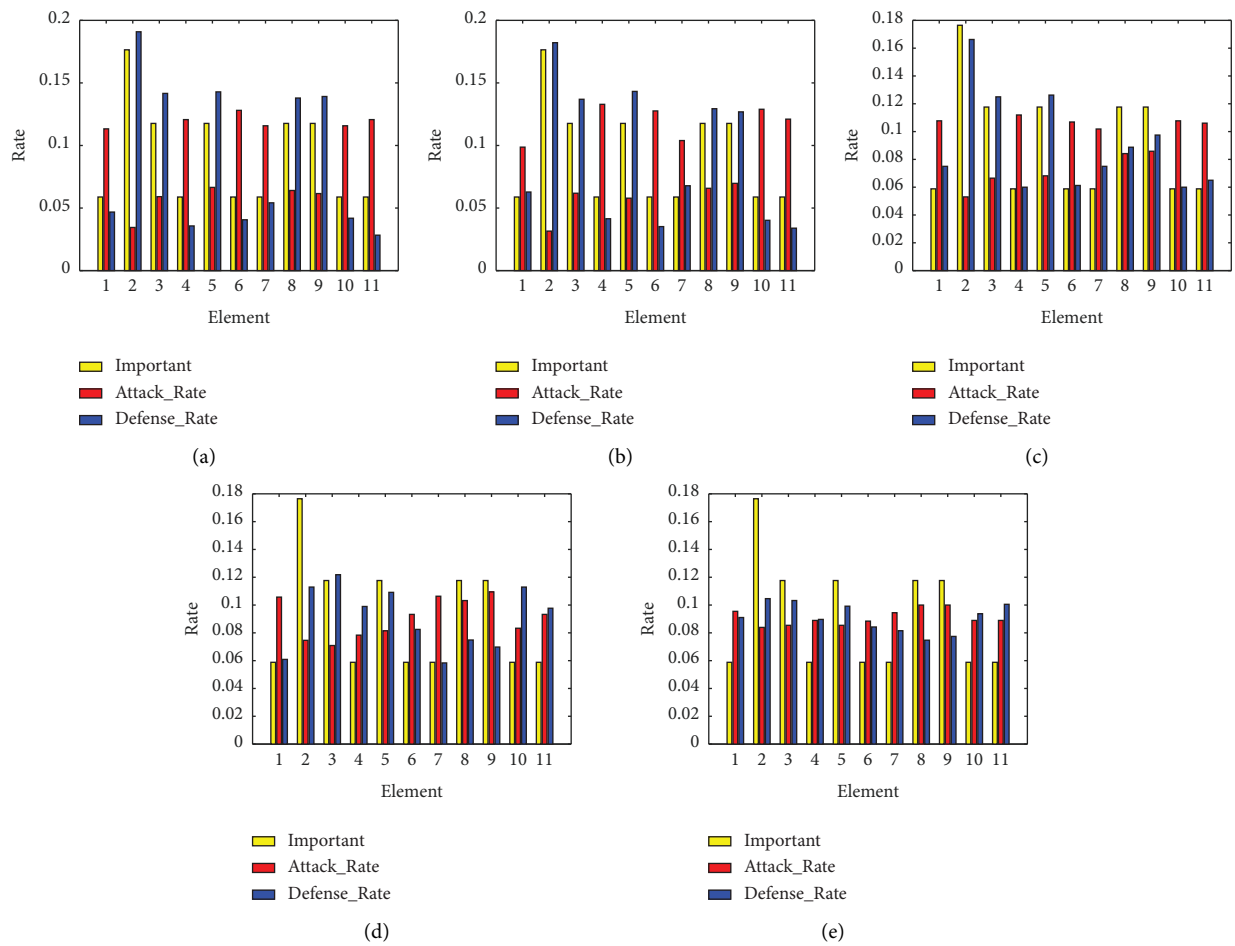


Figure 6: Experimental results under different attack resources. (a) Attack resources = 2. (b) Attack resources = 4. (c) Attack resources = 6. (d) Attack resources = 8. (e) Attack resources = 10.

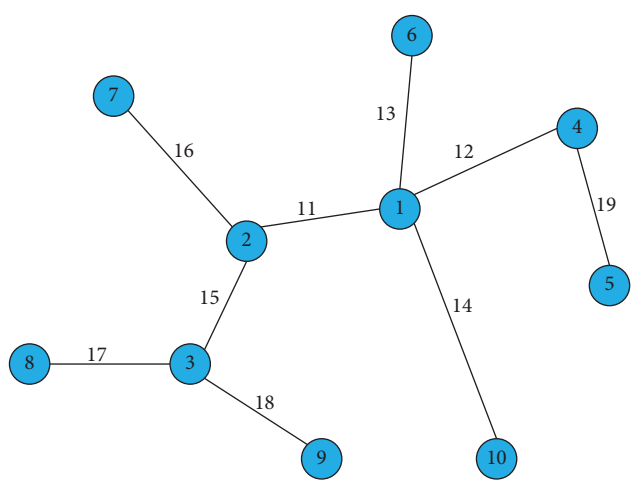


Figure 7: The graph of this simulation experiment.

also distributes defense resources equally to defense each element.

5.6. Strategies of Attacker and Defender with Different Attack Resources in Another Network. In this subsection, the network scale is expanded, and a simulation experiment with 10 nodes and 19 elements is carried out. The experimental diagram is shown in Figure 7. In this experiment, the defense resources are set to 15, and the attack resources are set to 5, 10, and 15, respectively. When the attack resources are set to 5, 10, and 15, the number of strategies from attacker is shown in Table 8. Compared with the network graph in Figure 4, although Figure 8 only

Table 8: The number of strategies under different attack resources.

	5	10	15
The number of strategies	$C_{19}^{5} = 11628$	$C_{19}^{10} = 92378$	$C_{19}^{15} = 3876$

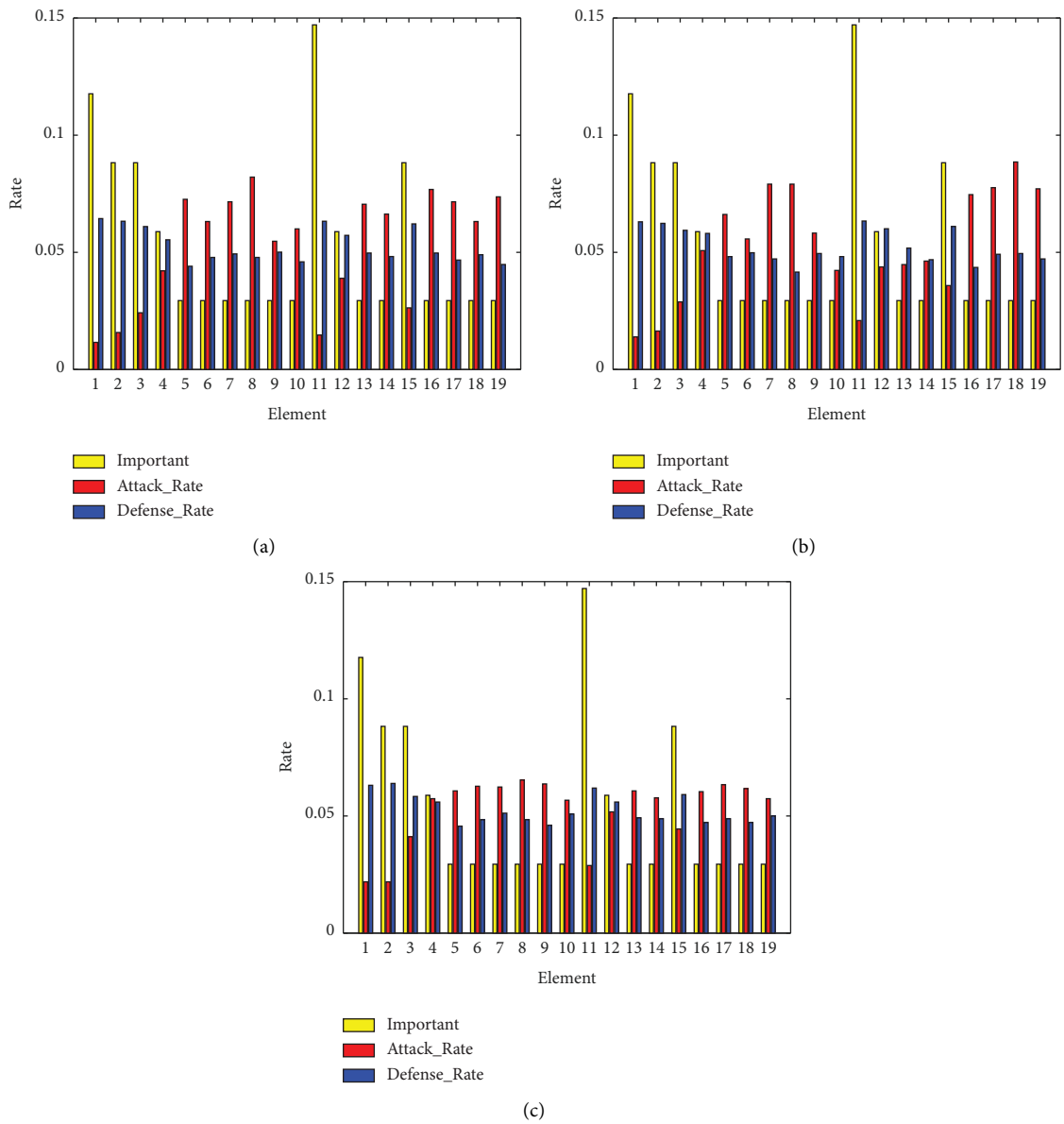


Figure 8: Experimental results under different attack resources. (a) Attack resources = 5. (b) Attack resources = 10. (c) Attack resources = 15.

increases a little network size, the attack and defense strategies increase a lot. The ncPSO is used to solve the Nash equilibrium of the game, and the probability of each element being attacked and defended is calculated. The average of 10 experimental results is taken to ensure the reliability of the experiment. The experimental results are shown in Figure 8.

As can be seen from the figure, the importance of the communication link cannot be ignored, and the importance of the communication link is not lower than that of the

communication equipment. Figure 8 shows that when the defense resource is 15 and the attack resource is 5, the defender will allocate resources evenly to deal with the attacks of attacker, and the attacker predicts that defender will allocates more resources to protect more important elements, so attacker will attack less important elements to obtain profit. When the defense resource is 15 and the attack resource is 15, the defender will select elements with higher importance to strengthen protection to reduce losses, and the attacker chooses the element that the defender allocates less defense resources to attack to maximize the benefit of benefit (see Table 9).

6. Conclusion

5G heterogeneous network of CPS is a more intelligent and open network system based on software defined network (SDN) and network function virtualization (NFV). The macro analysis of hacking behavior through game theory is an effective method to prevent hacking attacks. Therefore, we established a system model and a game model based on 5G heterogeneous networks of CPS, respectively. The solution of mixed-strategy Nash equilibrium in game models is the most important part of analyzing attack and defense behaviors. A ncPSO is proposed to solve the mixed-strategy Nash equilibrium and analyze the strategy choices of attacker and defender under different attack and defense resources through simulation experiments. Experiments demonstrate the effectiveness and superiority of the proposed ncPSO. The research in this paper provides a feasible macro analysis for defenders to defend against attacks of hacker.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This project was funded by the National Natural Science Foundation of China, under No. 61932005.

References

[1] S. Wijethilaka, M. Wijethilaka, and M. Liyanage, "Survey on network slicing for internet of things realization in 5g networks," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 957–994, 2021.

[2] F Qamar, M. H. D. N. Hindia, K Dimyati et al., "Interference management issues for the future 5g network: a review," *Telecommunication Systems*, vol. 71, no. 4, pp. 627–643, 2019.

[3] H. Hui, Y. Ding, Q Shi, F Li, Y. Song, and J. Yan, "5g network-based internet of things for demand response in smart grid: 5G network-based Internet of Things for demand response in smart grid: A survey on application potential survey on application potential," *Applied Energy*, vol. 257, Article ID 113972, 2020.

[4] S. J. Yang, Y. M. Pan, L. Y. Shi et al., "Millimeter-wave dual-polarized filtering antenna for 5g application," *IEEE Transactions on Antennas and Propagation*, vol. 68, no. 7, pp. 5114–5121, 2020.

[5] M. Zhao, "Research on the effect of iot wireless network technology on the educational management of Research on the Effect of IOT Wireless Network Technology on the Educational Management of China's Universitieshina's universities," *Security and Communication Networks*, vol. 2022, Article ID 9405897, 9 pages, 2022.

[6] M. A. Ouamri, M. E. Oteşteanu, A Isar, M.-E. Azni, A. Isar, and A. Mohamed, "Coverage, handoff and cost optimization for 5g heterogeneous network," *Physical communication*, vol. 39, Article ID 101037, 2020.

[7] Y. Xu, G. Gui, H. Gacanin, F. Adachi, H. Gacanin, and F. Adachi, "A survey on resource allocation for 5g heterogeneous networks: A Survey on Resource Allocation for 5G Heterogeneous Networks: Current Research, Future Trends, and Challengesurrent research, future trends, and challenges," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 668–695, 2021.

[8] J. Arroyo, A. Athreya, J. Cape et al., "Inference for multiple heterogeneous networks with a common invariant subspace," *Journal of machine learning research: JMLR*, vol. 22, no. 141, pp. 1–49, 2021.

[9] M. Krishna, S. Mohan Babu Chowdary, P. Nancy, and V. Arulkumar, "A survey on multimedia analytics in security systems of cyber physical systems and iot," in *Proceedings of the 2021 2nd International Conference on Smart Electronics and Communication (ICOSEC)*, pp. 1–7, IEEE, Trichy, India, October 2021.

[10] D. Almeida, L. F. Da Rosa Righi, R. Rodrigues et al., "Cyber-physical systems architectures for industrial internet of things applications in industry 4.0: Cyber-physical systems architectures for industrial internet of things applications in Industry 4.0: A literature review literature review," *Journal of Manufacturing Systems*, vol. 58, pp. 176–192, 2021.

[11] M. A. Hasnat, S. T. Ahmed Rumee, M. A. Razzaque, and M. Mamun-Or-Rashid, "Security study of 5g heterogeneous network: current solutions, limitations & future direction," in *Proceedings of the 2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, pp. 1–4, IEEE, Cox'sBazar, Bangladesh, February 2019.

[12] Y. Wu, A. Khisti, C. Xiao et al., "A survey of physical layer security techniques for 5g wireless networks and challenges ahead," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 4, pp. 679–695, 2018.

[13] B. Jia, X. Zhang, J. Liu et al., "Blockchain-enabled federated learning data protection aggregation scheme with differential privacy and homomorphic encryption in iiot," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 6, pp. 4049–4058, 2022.

[14] M. Min, L. Xiao, C. Xie et al., "Defense against advanced persistent threats in dynamic cloud storage: Defense Against Advanced Persistent Threats in Dynamic Cloud Storage: A Colonel Blotto Game Approach colonel blotto game approach," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4250–4261, 2018.

[15] K. Wang, L. Yuan, T. Miyazaki et al., "Jamming and eavesdropping defense in green cyber–physical transportation systems using a stackelberg game," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 9, pp. 4232–4242, 2018.

[16] L. Xinlong, C. Zhibin, and Z. Chen, "Ddos attack detection by hybrid deep learning methodologies," *Security and*

Communication Networks, vol. 2022, Article ID 7866096, 7 pages, 2022.

[17] F. Chaoqi, G. Yangjun, Z. Jilong, S. Yun, Z. Pengtao, and W. Tao, "Attack-defense game for critical infrastructure considering the cascade effect," *Reliability Engineering & System Safety*, vol. 216, Article ID 107958, 2021.

[18] H.-W. Zhang, L. I. Tao, and S.-R. Huang, "Network defense decision-making method based on attack-defense differential game," *ACTA ELECTONICA SINICA*, vol. 46, no. 6, p. 1428, 2018.

[19] B. Schneier, "Artificial intelligence and the attack/defense balance," *IEEE security & privacy*, vol. 16, no. 2, p. 96, 2018.

[20] H. Wu, J. Fan, C. Lai, and J. Liu, "Website defense strategy selection method based on attack-defense game and Monte Carlo simulation," *Journal on Communications*, vol. 39, no. 8, p. 48, 2018.

[21] Y. Li, Y. Deng, Y. Xiao, J. Wu, Y. Xiao, and J. Wu, "Attack and defense strategies in complex networks based on game theory," *Journal of Systems Science and Complexity*, vol. 32, no. 6, pp. 1630–1640, 2019.

[22] L. Leneutre, J. Chen, and L. Jean, "A game theoretical framework on intrusion detection in heterogeneous networks," *IEEE Transactions on Information Forensics and Security*, vol. 4, no. 2, pp. 165–178, 2009.

[23] X. Liu, B. Cui, J. Fu et al., "Secure data publishing of private trajectory in edge computing of iot," *Security and Communication Networks*, vol. 2022, Article ID 2045586, 14 pages, 2022.

[24] Q. Xu, Z. Su, Q. Zheng, M. Luo, B. Dong, and K. Zhang, "Game theoretical secure caching scheme in multihoming edge computing-enabled heterogeneous networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4536–4546, 2019.

[25] Y. Sun, Z. Tian, M. Li et al., "Automated attack and defense framework toward 5g security," *IEEE Network*, vol. 34, no. 5, pp. 247–253, 2020.

[26] T. Olivier, Y. Hayel, C. Kamhoua, and D. Gabriel, "Game theoretic modeling of cyber deception against epidemic botnets in internet of things," *IEEE Internet of Things Journal*, vol. 9, 2021.

[27] R. O. O. Adeogun, "A novel game theoretic method for efficient downlink resource allocation in dual band 5g heterogeneous network," *Wireless Personal Communications*, vol. 101, no. 1, pp. 119–141, 2018.

[28] Z.-L. Chang, C.-Y. Lee, C.-H. Lin, C.-Y. Wang, and H.-Y. Wei, "Game-theoretic intrusion prevention system deployment for mobile edge computing," in *Proceedings of the 2021 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, IEEE, Madrid, Spain, December 2021.

[29] F. He, J. Zhuang, N. S. V. Rao, J. Zhuang, and N. S. V. Rao, "Discrete game-theoretic analysis of defense in correlated cyber-physical systems," *Annals of Operations Research*, vol. 294, no. 1-2, pp. 741–767, 2020.

[30] M. Kim, "Game theoretic approach of eavesdropping attack in millimeter-wave-based wpans with directional antennas," *Wireless Networks*, vol. 25, no. 6, pp. 3205–3222, 2019.

[31] Y. Li, Y. Xiao, Y. Li, J. Wu, Y. Li, and J. Wu, "Which targets to protect in critical infrastructures-a game-theoretic solution from a network science perspective," *IEEE Access*, vol. 6, pp. 56214–56221, 2018.

[32] W. Jiang, B. X. Fang, Z. H. Tian, H. L. Zhang, Z. Tian, and H. Zhang, "Evaluating Network Security and Optimal Active Defense Based on Attack-Defense Game Model," *Chinese Journal of Computers*, vol. 32, no. 4, pp. 817–827, 2009.

[33] Q. Leng, Y. Yang, R. Pan, H. Yang, R. Pan, and H. Hu, "Research of complete information static game model for software manufacturer, white hats and black hats," *Procedia Computer Science*, vol. 131, pp. 832–840, 2018.

[34] A. Attiah, M. Chatterjee, and C. C. Zou, "A game theoretic approach to model cyber attack and defense strategies," in *Proceedings of the 2018 IEEE International Conference on Communications (ICC)*, pp. 1–7, IEEE, Kansas, MO, USA, May 2018.

[35] W. Wang and B. Zeng, "A two-stage deception game for network defense," in *Proceedings of the International conference on decision and game theory for security*, pp. 569–582, Springer, Seattle, WA, USA, October 2018.

[36] K. Ferguson-Walter, S. Fugate, J. Mauger, and M. Major, "Game theory for adaptive defensive cyber deception," in *Proceedings of the 6th Annual Symposium on Hot Topics in the Science of Security*, pp. 1–8, Nashville, TN, USA, April 2019.

[37] S. Shen, Y. Li, H. Xu, Q. Cao, H. Xu, and Q. Cao, "Signaling game based strategy of intrusion detection in wireless sensor networks," *Computers & Mathematics with Applications*, vol. 62, no. 6, pp. 2404–2416, 2011.

[38] J. Kennedy and E. Russell, "Particle swarm optimization,"vol. 4, pp. 1942–1948, in *Proceedings of the ICNN'95-international conference on neural networks*, vol. 4, pp. 1942–1948, IEEE, Perth, WA, Australia, November 1995.

[39] J. R. Sampson, *Adaptation in Natural and Artificial Systems*, MIT Press, Cambridge, MA, USA, 1976.

[40] S. Mirjalili, A. Lewis, and A. Lewis, "The whale optimization algorithm," *Advances in Engineering Software*, vol. 95, pp. 51–67, 2016.

[41] A. Hatamlou, "Black hole: Black hole: A new heuristic optimization approach for data clustering new heuristic optimization approach for data clustering," *Information Sciences*, vol. 222, pp. 175–184, 2013.

[42] X. S. Yang, X.-S. He, and X. He, "Bat algorithm: literature review and applications," *International Journal of Bio-Inspired Computation*, vol. 5, no. 3, pp. 141–149, 2013.

[43] S. M. Mirjalili, S. Z. Mirjalili, S. Saremi, and S. Mirjalili, "Sine cosine algorithm: theory, literature review, and application in designing bend photonic crystal waveguides," *Nature-inspired optimizers*, vol. 811, pp. 201–217, 2020.

[44] C. Y.-Y. Lee, "Mixed-strategy nash equilibrium in data envelopment analysis," *European Journal of Operational Research*, vol. 266, no. 3, pp. 1013–1024, 2018.

[45] H. J. Hilhorst, C.-J. Appert-Rolland, and C. Appert-Rolland, "Mixed-strategy nash equilibrium for a discontinuous symmetric n-player game," *Journal of Physics A: Mathematical and Theoretical*, vol. 51, no. 9, Article ID 095001, 2018.

[46] F. Neri, E. Mininno, G. Iacca, E. Mininno, and G. Iacca, "Compact particle swarm optimization," *Information Sciences*, vol. 239, pp. 96–121, 2013.

[47] B. C. Arnold, *Pareto Distribution*, pp. 1–10, Wiley, Hoboken, NJ, USA, 2014.

Hindawi
Security and Communication Networks
Volume 2022, Article ID 1317626, 11 pages
https://doi.org/10.1155/2022/1317626

Research Article

Security Enhancements for Data-Driven Systems: A Blockchain-Based Trustworthy Data Sharing Scheme

Yanping Wang,[1] **Xiaosong Zhang**,[1] **Xiaofen Wang**,[1] **Teng Hu**,[2] **Peng Lu**,[2] **and Mingyong Yin**[2]

[1]*UESTC, Institute for Cyber Security, School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China*
[2]*CAEP, Institute of Computer Application, China Academy of Engineering Physics, Mianyang, China*

Correspondence should be addressed to Teng Hu; mailhuteng@gmail.com

Received 9 August 2022; Accepted 14 September 2022; Published 11 October 2022

Academic Editor: Yuanyuan Huang

With the increasingly prominent value of big data, data sharing within enterprises and organizations has become increasingly popular, and many institutions have established data centers to achieve effective data storage and sharing. Meanwhile, cyberspace data security and privacy have become the most critical issue that people are concerned about since shared data often involves commercial secrets and sensitive information. At present, data encryption techniques have been applied to protect the security of the sensitive data stored in and shared by the data centers. However, the challenges of efficient data sharing, secure management of decryption keys, deduplication of the plaintext, and transparency and auditability of the data access arise. These challenges may obstruct the development of data sharing in data-driven systems. To meet these challenges, we propose a secure and trustworthy data sharing scheme and introduce blockchain, proxy re-encryption (PRE), and trusted execution environments (TEEs) into the data-driven systems. Our scheme mainly enables (1) automatic distribution and management of the decryption keys, (2) reduction of the reduplicative data, and (3) trustworthy data sharing and recording. Finally, we implement the proposed scheme and compare it with other existing schemes. It is demonstrated that our scheme reduces the computation and communication overhead.

1. Introduction

With the development of big data, the Internet of Things, and other network technologies, various kinds of data have been produced. The economic and social benefits of the data trigger the demand for sharing data between institutes and enterprises. Therefore, many organizations have established data centers utilizing the private or public cloud to realize effective data storage and sharing. Because data often involves business secrets and sensitive information, among others, data privacy and security are the key issues that people are concerned about, especially in large enterprises and scientific research institutes.

Encryption can be applied to protect the privacy and security of the sensitive data stored in and shared by the data center to a certain extent. Encrypted data sharing schemes [1, 2] are proposed, in which the data are encrypted by the owner and can only be decrypted by authorized users. In these scenarios [1, 2], an owner negotiates a session key with a group of users in advance so that they can share data with them. However, if a new user is added to the authorized sharing group, a new session key is needed to be negotiated, and data are required to be encrypted using this new session key. This inevitably introduces a large computation overhead if there are frequent changes in the sharing group.

In order to alleviate the above complex key management problems, the proxy re-encryption (PRE) techniques [3–5], which allow a proxy [e.g., cloud server (CS)] to convert a cipher of a delegator to different ciphers for different delegatees, have been used to share the data to different users dynamically without the complex key agreement and decrypt-then-encrypt operations. PRE properties make it a

practical approach to cloud-assisted data sharing. However, in order to avoid huge computation in PRE's cipher conversion, the CS may not generate the re-encryption ciphertext honestly [6]. Additionally, many PRE-based data sharing schemes [7, 8], cannot satisfy nonframeability. In other words, in these schemes, the CS may be maliciously framed for refusing to perform a re-encryption operation or for outputting a wrong reencrypted cipher when it indeed performs honestly.

Recently, blockchain has been applied in data sharing solutions [2, 9, 10], in which the encrypted data were stored in the off-chain data center (e.g., cloud), whereas the meta and data transfer log were recorded in the blockchain for data retrieval and auditing. Then, the data sharing schemes that combine the blockchain and PRE emerged such as in [11–14], where the encrypted data can be accessed by the authorized users with the help of a proxy server, and the misbehavior of the proxy server or the users in re-encryption can be hindered as all operations would be recorded in the blockchain and can be audited. Nevertheless, references [11–14] faced the efficiency challenge caused by the complex encryption/decryption and frequent interaction of data owners (DO) and users. On the one hand, frequent interactions and complex ciphertext transformations are heavy burdens to DO and CSs. On the other hand, the storage of large-size data is a great burden to the blockchain. In addition, in [11–14], the same data will be packaged into different ciphertexts, and the redundant plaintext copies would result in additional storage overhead. These challenges motivate us to propose a more efficient and versatile encrypted data sharing scheme.

In this paper, we combine blockchain, PRE, and trusted execution environments (TEEs) and propose a flexible and secure data sharing method for data-driven systems. By employing the PRE technology, our scheme allows the encrypted data to be transformed (by the CS) into different ciphertexts for different authorized users without the complex key agreement. Furthermore, by involving the blockchain, the misbehavior of the CS or the users in re-encryption can be recorded and audited. Meanwhile, the smart contract can automatically delegate the re-encryption key to authorized users so that the DO's computation and communication burden can also be reduced. Finally, by utilizing the TEEs, the smart contract can be executed in a secure enclave to protect the DO's private key. The major contributions of our scheme are as follows:

(1) Smart contract is employed to control the access of data, such that the decryption key's delegation can be executed automatically and the DO is not required to be online all the time, which greatly reduces the computation and communication burden of the DO and makes the data sharing convenient.

(2) Our scheme utilizes the tamper-proof and consensus properties of the blockchain, and the transfer logs of data requests and replays are recorded in the distributed ledger, which realizes the trustful recording and the real-time monitoring.

(3) In our scheme, the duplicate data can be quickly detected to avoid redundancy, and its storage request will be refused. Therefore, the ciphertexts corresponding to the same plaintext can be reduced.

(4) We conduct experiments to evaluate the performance of our proposed scheme, and the results show that our scheme is more efficient than the existing schemes [11–14] with respect to computation overhead and cipher size.

The rest of this paper is organized as follows: Section 2 surveys the related works. The preliminaries are presented in Section 3. The system model and design goals are described in Section 4. Section 5 introduces the detailed proposal, including the data release and retrieval. The analysis and simulation of the proposed solution are shown in Section 6. Finally, in Section 7, we draw our conclusion.

2. Related Works

With the rapid development of blockchain technologies, the schemes [2, 9, 15, 16] of decoupling the storage layer and the blockchain have been proposed to achieve efficient and reliable data sharing, especially in large-scale data-driven systems. In these cases, the data generated from the source are stored in the off-chain data center, and the meta (e.g., digest) is recorded on the blockchain for efficient data retrieval. When a user queries data from the blockchain, the distributed ledger's retrieval mechanism can help users quickly retrieve the queried information from the blockchain, which greatly improves the efficiency and credibility of the system. However, most of these schemes use blockchain just as a distributed and immutable database, and the issues such as trusted access control have not been completely solved.

The concept of PRE was initially introduced and constructed in [3], in which the DO controls the delegation of data access with the help of a CS. Based on this concept, Ran and Susan [4] proposed a secure PRE scheme against chosen ciphertext attacks. Next, Weng et al. [5] proposed a conditional PRE, which achieves a more fine-grained delegation. In order to ensure secure and efficient data sharing, PRE technology is used in [11–14] to realize multisharing controls of ciphertext for blockchain-based big data storage. In schemes [11–14], DO outsource their encrypted data to the cloud using identity-based encryption and grant legitimate users access to the data. However, these schemes face heavy communication and computation costs. Additionally, schemes [11–14] either rely on a proxy to fully manage their data or require the DO to always be online to delegate the decryption key to the data user, which means either the data transparency and auditability cannot be achieved or the DO needs to be online all the time.

For achieving privacy-preserving and automatic data sharing, Li et al. [17] proposed a blockchain-based privacy-preserving data sharing scheme with rewards, which uses smart contracts to automatically generate the decryption keys for users, and the TEEs are used to ensure the security of secret keys in smart contracts. Wang et al. [18] proposed a

TEEs-based smart contract execution scheme, which is used to share private data with fine-grained access control for smart grids. Lei et al. [19] proposed a multiparty data sharing platform that combines blockchain and TEEs and realizes automated data sharing. However, these schemes face communication and computation burdens caused by fine-grained access control.

Based on the above research, we provide a trust and efficient data sharing scheme. We incorporate blockchain, TEEs, and proxy re-encryption to achieve secure data sharing while achieving (1) efficient one-to-many sharing of data, (2) automatic distribution and management of the decryption keys, (3) reduced reduplicative data storage, and (4) trustworthy data transmission and records.

3. Preliminaries

This section briefly outlines the preliminaries about pairing groups, blockchain technologies, and PRE. The key notations involved in this paper are summarized in Table 1.

Table 1: Key notions.

Notions	Description
$\|$	Data concatenation
h, h_1, H	Cryptographic hash functions
$(sk_{\text{tee}}, pk_{\text{tee}})$	Private/public key pair of a TEE
(sk_o, pk_o)	Private/public key pair of DO
(sk_c, pk_c)	Private/public key pair of DC
(sk_s, pk_s)	Private/public key pair of CS
$(dk\ , ek)$	Randomly generated key pair by DO
$SEnc(\cdot), S\ De\ c(\cdot)$	Symmetric encryption and decryption
$AEnc(\cdot), A\ De\ c(\cdot)$	Asymmetric encryption and decryption
$sig_{sk_i}(\cdot)$	Signature under the secret key sk_i
k_{st}	State key of smart contracts
$k_{o\longrightarrow c}$	Reencryption key for DC
CF, CF'	Original ciphertext and reencrypted ciphertext

3.1. Pairing Groups. Let $pp = (p, \mathbb{G}, \mathbb{G}_T, e)$ be the pairing parameter, where $\mathbb{G}, \mathbb{G}_T$ is the finite groups of order p and e an efficiently computable bilinear map from $\mathbb{G} \times \mathbb{G}$ to $\mathbb{G}_T$, which satisfies the following:

(1) Bilinearity: for any generator $P, Q \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, the equation $e(aP, bQ) = e(P, Q)^{ab}$ holds;

(2) Nondegenerability: $e(P, Q) \neq 1$;

(3) Computability: e can be efficiently computed.

Difficult problems based on the above bilinear pairings are defined as follows:

Definition 1. (DL assumption) [3]. Let $\mathbb{G}$ be a cyclic group. The Discrete Logarithm (DL) assumption is that, for all $P \in \mathbb{G}$ and $a \in \mathbb{Z}_p$, given an input $\{P, aP\}$, the probability of outputting a is negligible for any polynomial time algorithm.

Definition 2. (3-QBDH assumption) [5]. Let $pp = (p, \mathbb{G}, \mathbb{G}_T, e)$ be the pairing parameter and P a generator of $\mathbb{G}$. The 3-quotient bilinear Diffie–Hellman (DBDH) assumption on pp is as follows: for any unknown $a, b \in \mathbb{Z}_p$, given $\{P, -aP, aP, a^2P, bP\}$, the probability of computing $e(P, P)^{b/a^2}$ is negligible for any polynomial time algorithm.

3.2. Some Basic Knowledge of Blockchain. With the launch of the bitcoin network [20], the concept of blockchain has become widely known to the public. As a decentralized distributed ledger maintained by multiple parties, the primary purpose of blockchain is to solve the trust problem in untrustworthy distributed environments by using peer-to-peer (P2P) network schemes, consensus algorithms, asymmetric encryption, password hashing, and other technologies. The blockchain can be used as a secure data management system [15] to ensure data integrity and availability. It can also be used as a supervision and audit platform [21] to achieve transparent supervision of the data. Additionally, the blockchain can be used as a platform [17] that achieves secure and trusted data processing by utilizing smart contracts (self-executing programs with clauses clearly specified by the underlying code and deployed on the blockchain).

3.3. Trusted Execution Environments. As data in smart contracts are transparent on the blockchain, users' private information can easily be exposed [17]. TEEs, such as Intel Software Guard Extensions (SGX) [22], TrustZone [23], and MultiZone [24], can be utilized to solve this problem. The TEEs enable secure execution of programs on untrusted hosts (e.g., cloud), as the programs can be run in a protected manner by isolating all the operations against the outside world. They also allow remote verifiers to ascertain a device's current configuration and behavior *via* remote attestation. It is worth noting that, *via* remote attestation, a TEE can build a secure channel for the user and other TEEs to communicate with it securely. These properties make TEEs a good choice for processing and sharing private data. For example, Bowman et al. [25] proposed a TEEs-based private data process scheme, named private data objects (PDOs), which allows mutually untrusted parties to work on private data based on preagreed policies, and the open-source code is provided in [26].

3.4. Proxy Re-encryption. The concept of the RRE was introduced by Blaze et al. [3], in which a semitrusted proxy server is delegated to convert a delegator's ciphertext to a delegatee's without the leakage of the corresponding plaintext. The PRE can be used for secure data sharing in cloud environments, which usually consists of the following five algorithms:

$\mathbf{KeyGen}(\lambda) \longrightarrow (\mathbf{pk}, \mathbf{sk})$: on the input of the security parameter λ, this algorithm outputs a public/private key pair (pk, sk);

$\mathbf{Re-Key}(\mathbf{sk_o}, \mathbf{pk_c}) \longrightarrow \mathbf{r_{o\longrightarrow c}}$: on the input of a user's private key sk_o and another user's public key pk_c, this algorithm outputs a re-encryption key $r_{o\longrightarrow c}$;

Encryption $(\mathbf{pk_o}, \mathbf{M}) \longrightarrow \mathbf{CF}$: on the input of a user's public key pk_o and the plaintext M, this algorithm outputs a ciphertext CF under the public key pk_o;

Re – Encryption $(\mathbf{CF}, \mathbf{r_{o\longrightarrow c}}) \longrightarrow \mathbf{CF'}$: on the input of the ciphertext CF under the public key pk_o and the re-encryption key $r_{o\longrightarrow c}$, this algorithm outputs a ciphertext CF' under the public key pk_c;

De cryption $(\mathbf{sk_c}, \mathbf{CF'}) \longrightarrow \mathbf{M}$: on the input of a user's secret key sk_c and the reencrypted ciphertext $CF\prime$, this algorithm outputs a plaintext M.

4. System Model

In this section, we illustrate the framework, outline the threat model, and design goals of the proposed scheme.

4.1. Framework. The schematic diagram of our framework is shown in Figure 1, in which the consensus is separated from the execution of the smart contract. Similar to Ekiden [27], our framework consists of a CS, consensus nodes, participant nodes, and authorities. Each component and its role are described below.

4.1.1. Cloud Server. It is usually a data center responsible for storing and securely sharing the encrypted data for the users with the help of a TEE (this TEE in the cloud is called sTEE). The CS loads the smart contract, executes it in the sTEE to generate a re-encryption key, and performs the reencrypting operations.

4.1.2. Consensus Nodes. There are two types of consensus nodes: without TEEs and with TEEs. A consensus node without TEEs is responsible for maintaining the blockchain ledger and realizing basic blockchain functions such as packaging blocks and verifying blocks. Besides maintaining the ledger, consensus nodes with TEEs play the role of key management committees (these TEEs are called kTEEs) and are responsible for managing secret keys and remotely attesting to the sTEE.

4.1.3. Participating Nodes. They are the blockchain users, including the DO and data consumers (DC). DO is responsible for data release; DC requests the data by revoking the smart contract. After that, DC obtains the reencrypted data, which can be decrypted using their private key.

4.1.4. Authorities. There are two types of authorities, certificate authority (CA) and judgment authority (JA). The CA is responsible for membership enrollment and certificate distribution, and the JA is responsible for judging whether malicious behaviors have been performed.

Algorithm 1 further illustrates the interactions of Figure 1 among CS, consensus nodes, and participating nodes. Algorithm 1 shows that the interactions include two parts, data release and data retrieval, and DC can obtain data M when the algorithm ends.

4.2. Threat Model and Design Goals. In our system, the authorities are trusted, and DO will honestly share the ciphertext of the data. CS and some unauthorized DCs are curious about DO's data, and CS may not honestly transfer the cipher to DC. Moreover, the openness of blockchain enables the analysis of the transaction information (such as input and output) [28], which may cause the leakage of DO's data or DC's identity and attributes. We further assume the data and program can be securely stored and executed in the TEEs.

Based on the above security hypothesis, our goals are achieved if the below properties are satisfied.

4.2.1. Data Confidentiality. The DO's data should be kept confidential to CS during the storage and computation. Moreover, except for the DO and the authorized DC, other users cannot obtain the original data.

4.2.2. Nonredundant Storage. The storage requests of duplicate data should be quickly detected and refused for reducing storage costs.

4.2.3. Verifiable Integrity. When obtaining the data from CS, data integrity can be easily verified by DC.

4.2.4. Anonymity. The DC's identity and attributes should not be recognized by anyone during the data requesting process.

4.2.5. Transparency and Auditability. DO should know whom their data are shared with. Besides, the access and computation processes should be auditable.

4.2.6. Efficient Computation. The data encryption/decryption should avoid the heavy cryptographic overhead and save computation costs as much as possible.

5. The Proposed Approach

The proposed data sharing scheme includes three phases: system initialization, data release, and data retrieval. CA generates the system parameters, and each entity generates a private/public key pair and then registers to CA in the system phase. In the data release phase, DO releases their encrypted data and delegates the corresponding secret key shares to a group of kTEEs. Finally, in the data retrieval phase, DC invokes the smart contract and will obtain a re-encryption ciphertext, and then DC decrypts the re-encryption ciphertext to recover DO's data.

5.1. System Initialization. CA chooses a security parameter λ and generates the pairing parameter $pp = (p, \mathbb{G}, \mathbb{G}_T, e)$. CA also chooses a generator P in $\mathbb{G}$, a symmetric encryption algorithm *SEnc* (such as AES), an asymmetric encryption algorithm *AEnc* (such as ElGamal), and three secure cryptographic hash functions, h, h_1, and H, where h:

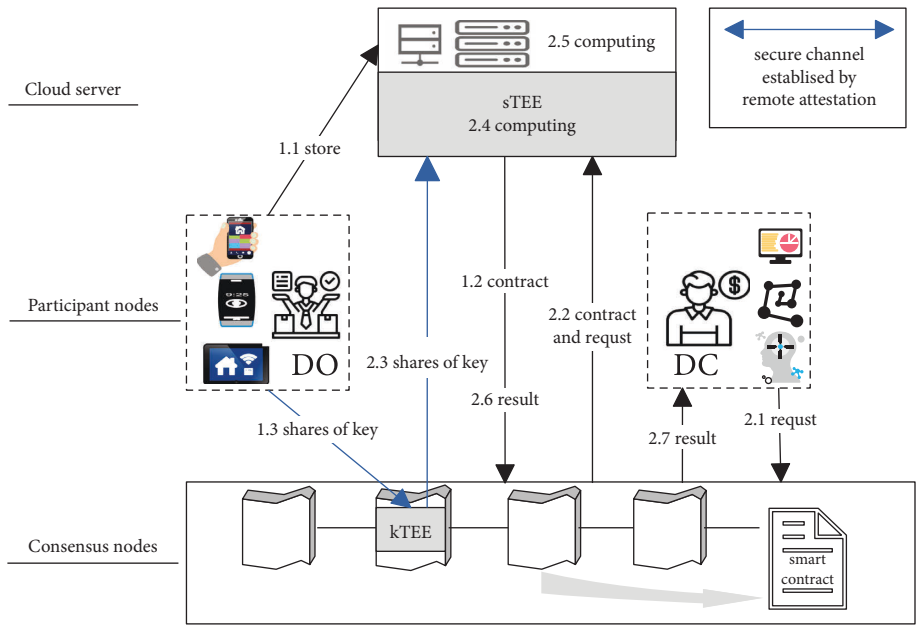


Figure 1: The proposed framework.

Require: DO: the encrypted data CF, the contract's program code Contract, the secret key shares $dk_1, dk_2, \ldots, dk_n$; DC: the request req.
Ensure: the data M.
procedure DATA RELEASE:
Step 1.1: DO sends CF and Contract to CS;
Step 1.2: CS publishes Contract to blockchain;
Step 1.3: DO checks Contract
if the Contract in the blockchain is correct, **then**
DO sends the secret key shares dk_i to the kTEE i.
procedure **Data retrieval:**
Step 2.1: DC revokes the smart contract with input req;
Step 2.2: CS loads Contract and req into the sTEE;
Step 2.3: sTEE performs remote attestation with kTEEs
if sTEE environment and loaded data are correct, **then**
The kTEE i transmits dk_i to the sTEE;
Step 2.4: CS executes the smart contract in the internal sTEE to obtain a reencryption key;
Step 2.5: CS computes the reencrypted ciphertext $CF\prime$ outside the sTEE;
Step2. 6: CS sends the reencrypted data $CF\prime$ to the blockchain;
Step 2.7: DC obtains $CF\prime$ and decrypts it to obtain M.

Algorithm 1: The process of the proposed scheme.

$\{0,1\}^{l_1} \leftarrow \{0,1\}^*$, h_1: $\mathbb{Z}_p \leftarrow \{0,1\}^*$, and H: $\{0,1\}^{l_2} \leftarrow \mathbb{G}_T$, and l_1 and l_2 represent the output lengths of hash. CA stores the system parameters $(pp, P, SEnc, AEnc, h, h_1, H)$ on the blockchain.

Each participant node i picks a private key $sk_i \in \mathbb{Z}_p$, computes the public key $pk_i = sk_i P$, and then registers to CA. CA then issues a certificate cert_i to user i. The certificate is combined with the user's public key pk_i and attributes.

The CS picks a private key $sk_s \in \mathbb{Z}_p$, computes the public key $pk_s = sk_s P$, and then registers to CA. CS and key management nodes initialize their TEEs and send the necessary information (e.g., the TEE's public key pk_{tee}) to the blockchain.

5.2. Data Release. In our scheme, data are encrypted and stored off-chain while the related information is stored on-chain, and the DO can specify whom their data can be shared with. The data release (Figure 2) is conducted as follows.

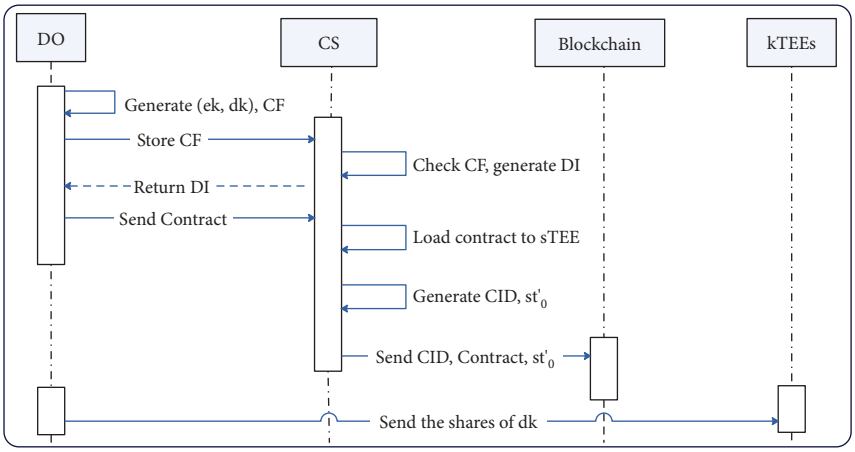


Figure 2: Sequence diagram of data release.

5.2.1. Data Encryption and Storage. DO randomly generates a private/public key pair (dk, ek) that satisfies $ek = dk \cdot P$. Then, DO randomly chooses $r \in \mathbb{Z}_p$ and computes $s = h(m)$, $C_0 = SEnc_s(m)$, $C_1 = r \cdot ek$, and $C_2 = s \oplus H(e(P,P)^r)$, where $SEnc_s$ represents the symmetric encryption under the secret key s. DO sends $CF = (C_0, C_1, C_2)$ to CS. If C_0 is not duplicated with other ciphers (this means m is different from other data), DO will receive a data retrieval index DI and a timestamp TS from the CS, where $DI = sig_{sk_s}(h(CF) \| \mathrm{TS})$. DO checks the validity of the DI using the CS's public key pk_s. If it is valid, then it continues; otherwise, it aborts.

5.2.2. Smart Contract Creation. The smart contract is responsible for generating the re-encryption key for the authorized DC, and the contract's creation is carried out as follows:

(1) DO creates a smart contract Contract, which is written in the form of program codes. Then, DO chooses a state key k_{st} and generates k'_{st} by encrypting k_{st} under the sTEE's public key pk_{tee}. DO sends the Contract to the CS. The Contract contains the related information $RI = \{ek, A, W, DI, k_{st}, TS, sig_{sk_o}(ek, A, W, DI, k'_{st}, TS)\}$, where A is the access attributes, W is the keywords set, k'_{st} is the encrypted state key, and $\mathrm{sig}_{sk_o}(\cdot)$ is the signature under DO's private key sk_o.

(2) CS loads the code of Contract into the sTEE, and then the sTEE generates a new contract ID, namely, CID, and decrypts k'_{st} using its secret key sk_{tee} to recover the state key k_{st}. Then, the sTEE encrypts the initial contract state as $st'_0 = SEnc_{k_{st}}(\overrightarrow{0})$ and outputs $\{\mathrm{CID}, \mathrm{Contract}, st'_0, \pi\}$, where π is a correctness proof generated using sTEE's secret key sk_{tee}. After that, CS sends the output to the blockchain. The consensus nodes will verify π, pack the legitimate $\{\mathrm{CID}, \mathrm{Contract}, st'_0, \pi\}$ into a block, and record it on the blockchain.

(3) After the Contract has been confirmed in the blockchain, DO sends CID and the shares of dk to the kTEEs. The dk is shared using the secret-sharing schemes [29, 30], and each share is encrypted using the corresponding kTEE's public key. The security feature of TEEs ensures that dk is kept secret against other nodes.

5.3. Data Retrieval. The anonymous data retrieval is conducted as shown in Figure 3, which is comprised of three phases: off-chain re-encryption key generation, cipher re-encryption, and data decryption. In the first phase, DC requests the cipher by invoking the smart contract, and then the smart contract executed in the sTEE generates a re-encryption key for the authorized DC. In the second phase, CS reencrypts the cipher and sends it to DC. In the last phase, DC receives the encrypted data and decrypts it to obtain the plaintext.

5.3.1. Off-Chain Re-encryption Key Generation. DC retrieves the interested keyword W_i on the blockchain and obtains CID from the related information RI. After checking the contents of the contract, DC invokes the smart contract with input $\mathrm{req} = \{\mathrm{CID}, AEnc_{ek}(DI, pk_c, \mathrm{cert}_c)\}$, where $AEnc_{ek}$ represents the asymmetric encryption under the public key ek. Then, the CS loads the corresponding contract state st'_{old} and req into sTEE, and the sTEE performs remote attestation with kTEEs to attest the sTEE environment. The loaded smart contract and data are correct. After passing the attestation, the shares of the decryption key dk will be transmitted to sTEE through secure channels. Once obtaining enough shares, the smart contract in sTEE performs the following steps:

(1) recovers the decryption key dk and state key k_{st};

(2) ecrypts st'_{old} using the state key k_{st} and obtains the old state st_{old}.

(3) ecrypts req using the decryption key dk and obtains the certificate of DC.

(4) hecks whether DC satisfies the access condition by verifying DC's attributes in their certificate. If it is satisfied, it continues. Otherwise, it jumps to step 7.

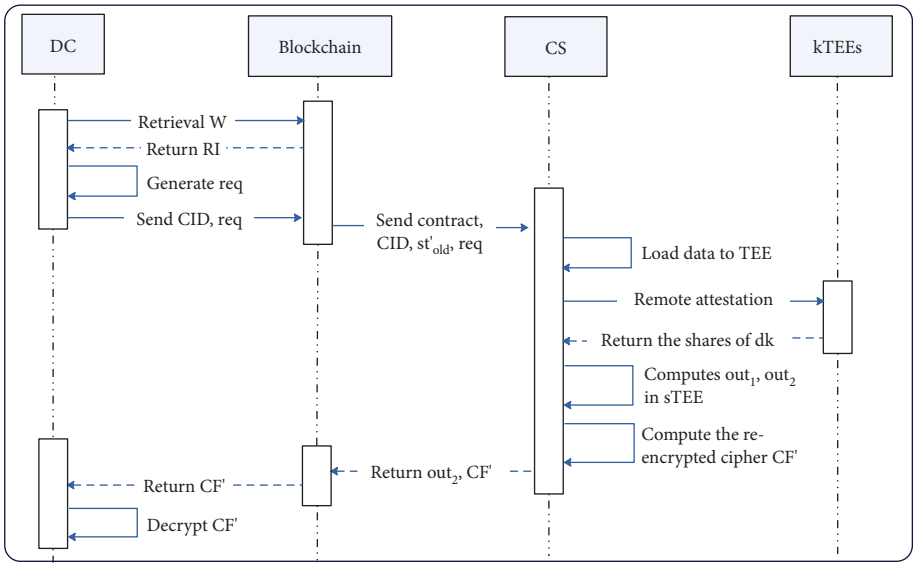


FIGURE 3: Sequence diagram of data retrieval.

(5) omputes $k = h_1(DI \| pk_s \| ek \| sk_{\text{tee}} \cdot pk_c)$.

(6) omputes a re-encryption key $k_{o \longrightarrow c} = (-\text{dk}) \cdot k \cdot pk_c$.

(7) pdates the contract state as st_{new} and computes $st'_{\text{new}} = SEnc_{k_{st}}(st_{\text{new}})$.

(8) utputs the executed results.

When the execution ends, all involved keys and intermediate results of the off-chain smart contract execution in sTEE can be securely erased [18]. There are two outputs: $\text{out}_1 = \{DI, k_{o \longrightarrow c}\}$ and $\text{out}_2 = \{st'_{\text{new}}, h(k_{o \longrightarrow c}), \pi\}$ (if DC is illegal, then $\text{out}_1 = \perp$ and $\text{out}_2 = \{st'_{\text{new}}, \perp, \pi\}$).

5.3.2. Cipher Re-encryption. CS retrieves the cipher CF according to DI and reencrypts CF to obtain $CF\prime = \{C'_0 = C_0, C'_1 = e(C_1, k_{o \longrightarrow c}), C'_2 = C_2\}$. CS then sends the reencrypted cipher to a temporary location, and a transaction $\text{tran} = \{\text{out}_2, \text{url}, h(CF\prime), \text{sig}_{sk_s}(\text{out}_2, \text{url}, h(CF\prime))\}$ is sent to the blockchain by CS, where url is the link of the temporary location that $CF\prime$ stores. For the transaction tran, consensus nodes check the validity of out_2 through the proof π provided by sTEE, check the validity of the url and $h(CF\prime)$ through the signature provided by CS, and maintain the consistency of state through consensus schemes.

5.3.3. Data Decryption. DC can retrieve the location url from the blockchain and download the cipher. After that, DC computes $k = h_1(DI \| pk_s \| ek \| sk_c \cdot pk_{\text{tee}})$. DC then decrypts the cipher $CF\prime$ by computing $s = C'_2 \oplus H(C'_1(-k) \cdot (-sk_c))$ and $m = S\ De\ c_s(C'_0)$, where $S\ De\ c_k$ is the symmetric decryption under the key s. DC verifies the data by checking if $h(m) \stackrel{?}{=} s$. If yes, DC accepts it; otherwise, DC rejects it and complains to the authority JA.

5.4. Claim. JA firstly requests the CS to provide the cipher CF and the sTEE's output out_1. After confirming the correctness of CF (using the index DI in the blockchain), JA computes the hash of the re-encryption key $H(k_{o \longrightarrow c})$ and compares if it is equal to that in out_2 of the blockchain. If yes, JA computes the re-encryption cipher $(\overline{C_0}, \overline{C_1}, \overline{C_2})$ and compares if $\overline{C_0} = C'_0$, $\overline{C_1} = C'_1$, and $\overline{C_2} = C'_2$ hold. If they hold, it ignores this complaint; otherwise, the CS has misbehaved, JA takes action accordingly.

6. Analysis and Evaluation

In this section, we analyze the security properties and evaluate the performance of the proposed scheme.

6.1. Security Analysis. Our scheme achieves the security properties of correctness, confidentiality, verifiable integrity, transparency, and auditability.

Theorem 1. *If DO, DC, and CS execute the scheme honestly, then DC can obtain DO's data correctly.*

We can prove Theorem 1 by verifying the following equation:

$$\begin{aligned} s &= C'_2 \oplus H(C'_1(-k) \cdot (-sk_c)) \\ &= s \oplus H(e(P,P)^r) \oplus H\left(e(C_1, k_{o \longrightarrow c})^{(-k)\cdot(-sk_c)}\right) \\ &= s \oplus H(e(P,P)^r) \oplus H\left(e(r \cdot dk \cdot P, (-dk) \cdot k \cdot pk_c)^{(-k)\cdot(-sk_c)}\right) \\ &= s \oplus H(e(P,P)^r) \oplus H(e(P,P)^r) = s. \end{aligned} \tag{1}$$

Theorem 2. *Our scheme achieves confidentiality if the 3-QBDH assumption holds.*

We prove the confidentiality of our scheme by proving that secret s cannot be recovered from the ciphertext by unauthorized users. We constructed an algorithm $\mathscr{B}$ that is given the pairing parameters $(\mathbb{G}, \mathbb{G}_T, p, e)$ and an instance $(P, A_0 = -aP, A_1 = aP, A_2 = (a^2)P, B = bP, T)$ and aimed to decide whether $T = e(P, P)^{b/a^2}$. $\mathscr{B}$ controls a hash oracle and runs an algorithm $\mathscr{A}$ (aimed to break the confidentiality of s) as a subroutine. We can prove that if $\mathscr{A}$ breaks the confidentiality of s, then $\mathscr{B}$ can break the 3-QBDH problem.

Before starting, we define two lists, $\mathscr{L}_h$ and $\mathscr{L}_c$, where $\mathscr{L}_h$ is the list of honest users and $\mathscr{L}_c$ is the list of corrupt users.

(i) Init phase: $\mathscr{A}$ prepares lists $\mathscr{L}_h$ and $\mathscr{L}_c$ and outputs $i^\star \in \mathscr{L}_h$ as the challenger user. Let sk_{i^*}, sk_i be the random numbers chosen from $\mathbb{Z}_p$. The public key for the challenge user $i^\star$ is set as $pk_{i^*} = sk_{i^*}A_2$ and the corresponding secret key is $a^2 sk_{i^*}$. Public keys of other honest users $i \in \mathscr{L}_h$ are set as $pk_i = sk_i A_1$, and the corresponding secret key is $a \cdot sk_i$. Public keys of corrupt users $i \in \mathscr{L}_c$ are $pk_i = sk_i P$, and the corresponding secret key is sk_i. It should be noted that the corrupt users' key pair $(sk_i, pk_i)_{i \in \mathscr{L}_c}$ is known as $\mathscr{A}$.

(ii) Find phase: $\mathscr{A}$ plays the role of user j and queries a re-encryption key of user i from $\mathscr{B}$. If $i = i^\star$ and $j \in \mathscr{L}_h$, $\mathscr{B}$ randomly chooses $k \in \mathbb{Z}_p$ and computes $k_{i \longrightarrow j} = sk_j \cdot k \cdot (-sk_{i^*}) \cdot A_0 = -(sk_{i^*}a^2) \cdot (sk_j a) \cdot k \cdot P$. If $i, j \neq i^\star$, $i \in \mathscr{L}_h$, and $j \in \mathscr{L}_h$, $\mathscr{B}$ randomly chooses $k \in \mathbb{Z}_p$ and computes $k_{i \longrightarrow j} = sk_j \cdot k \cdot (-sk_i) \cdot P = -(sk_i a) \cdot (sk_j a) \cdot k \cdot P$. If $i \in \mathscr{L}_h$, $i \neq i^\star$ and $j = i^\star$, $\mathscr{B}$ randomly chooses $k \in \mathbb{Z}_p$ and computes the rekey $k_{i \longrightarrow j} = sk_{i^*} \cdot k \cdot (-sk_j) \cdot A_1 = -(sk_j a) \cdot (sk_{i^*}a^2) \cdot k \cdot P$. If $i \neq i^\star$, $i \in \mathscr{L}_h$, and $j \in \mathscr{L}_c$, $\mathscr{B}$ randomly chooses $k \in \mathbb{Z}_p$ and computes $k_{i \longrightarrow j} = sk_j \cdot k \cdot (-sk_i) \cdot A_0 = -(sk_i a) \cdot sk_j \cdot k \cdot P$. If $i \in \mathscr{L}_c$, $\mathscr{B}$ computes $k_{i \longrightarrow j} = (-sk_i) \cdot k \cdot pk_j$.

(iii) Challenge phase: $\mathscr{A}$ chooses two numbers (s_0, s_1) and sends them to $\mathscr{B}$. $\mathscr{B}$ chooses s_b, where $b \in \{0, 1\}$. $\mathscr{B}$ sets the challenge cipher $C^\star$ as $C_1^\star = sk_{i^*}B$ and $C_2^\star = s_b \oplus H(T)$ and returns $C^\star$ to $\mathscr{A}$.

(iv) Guess phase: $\mathscr{A}$ outputs its decision of $s_{b\prime \in \{0,1\}}$. If $b\prime = b$, $\mathscr{B}$ outputs 1, which means $T = e(P, P)^{b/a^2}$ and outputs 0 otherwise, which means T is a random number of $\mathbb{G}_T$.

The confidentiality of secret s is converted to the hardness of the 3-QBDH problem. If T is a random number, then the probability of $\mathscr{A}$ to break the confidentiality of our scheme is 1/2. If $T = e(P, P)^{b/a^2}$, then $C^\star$ is a valid ciphertext of m_b with $r = b/a^2$, $pk_{i^*} = sk_{i^*}A_2 = sk_{i^*}a^2 P$, and $B = bP$. Therefore, if $\mathscr{A}$ can break the confidentiality of our scheme with advantage ε, then $\mathscr{B}$ can break the 3-QBDH assumption with advantage $1/2\varepsilon$.

Theorem 3. *Our scheme reduced the redundant storage, meanwhile, satisfied the verifiable integrity.*

In the data release phase, data m are encrypted under key s, equal to $h(m)$. Therefore, the same data m will be encrypted under the same symmetric key s, and the ciphertext $C_0 = SEnc_s(m)$ will be the same. Therefore, CS can quickly detect whether plaintext m has already existed in the database and refuse the redundant data's storage request. Furthermore, when DC decrypts key s from ciphertexts C_1', C_2' and then recovers data m' from C_0 using s, they can compare if $h(m\prime)$ equals s to verify the integrity. If the verification fails, DC can infer that the ciphertext has been modified or has not been generated correctly.

Theorem 4. *Our scheme satisfies the anonymity for DC.*

In the data retrieval phase, DC uses the regenerated pseudonym addresses to request the cipher. Meanwhile, the certificate is encrypted and can only be decrypted in the sTEE. Therefore, others cannot know the real identity and attributes of DC, and no one can recognize DC from the pseudonym. Therefore, our scheme achieves anonymous data retrieval.

Theorem 5. *Our scheme achieves transparency and auditability.*

The data access events are transparently recorded in the unforgeable ledger as transactions, publicly auditable to DO and JA. If the CS did not honestly generate the reencrypted cipher or if the CS was maliciously accused of returning the wrong reencrypted ciphertext, JA could easily detect it using the information in the ledger.

6.2. Performance Evaluation. We evaluate the computation and communication overheads of our scheme and then compare them with the related blockchain-based PRE schemes [11–14]. The symmetric pairings e: $\mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_T$ over the elliptic curve E: $y^2 = x^3 + 3x \bmod p$ with embedding degree 2 are constructed, the field size is 520 bits, and the group order is 160 bits. The bilinear pairing achieves the security level of 80 bits. Our simulations are supported by the MIRACL library [31], and our execution environment is performed on a laptop with AMD Ryzen 5 3550H 2.10 GHz processor and 16.00 GB memory.

6.2.1. Computation Overhead. The key cryptographic operations are Par, Exp, and Sm, which means the bilinear pairing operation, the exponentiation operation in $\mathbb{G}_T$, and the scalar multiplication operation in $\mathbb{G}$, respectively. Based on this setting, the main computational costs of our scheme are listed in Table 2. In the data release phase, DO performs 1 Sm operation and 1 Par. In the data retrieval phase, the smart contract performs 2 Sm operations, the CS performs 1 Par operation, and DC performs 1 Exp operation.

In order to demonstrate the efficiency of our scheme, we compare our scheme with the related schemes [11–14] in

Table 2: The computational complexity of each phase.

Phase	Entity	Operation
Data release	DO	Sm + Exp
Data retrieval	Smart contract	2 Sm
	CS	Par
	DC	Exp

Table 3: Computational comparison.

	Data release	Data retrieval
[11]	3 Sm + Exp	4 Sm + 2 Par
[12]	2 Sm + 2 Exp	8 Sm + 2 Par
[13]	2 Sm + Par	Sm + 2 Par + Hp
[14]	3 Sm + Par	3 Sm + 4 Par
Ours	Sm + Exp	Par + Exp

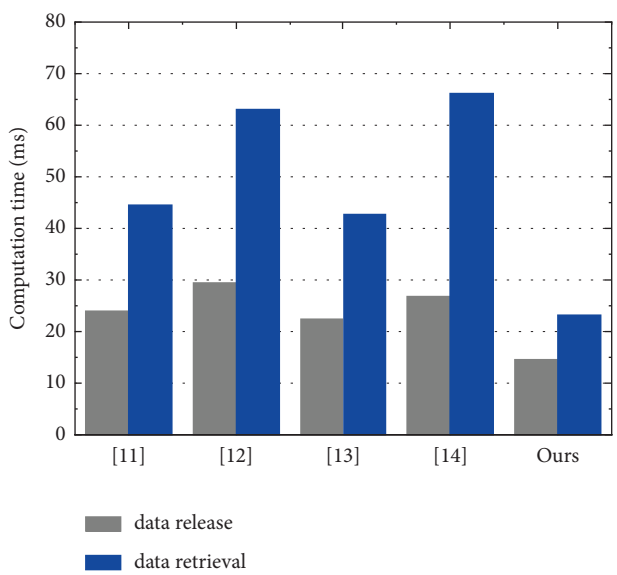


Figure 4: Computational overhead comparison.

Table 4: Communication overhead comparison.

	Ciphertext	Reencrypted ciphertext
[11]	$1\|G\| + 1\|G_T\|$	$4\|G\| + 1\|G_T\|$
[12]	$2\|G\| + 2\|G_T\|$	$1\|G\| + 2\|G_T\|$
[13]	$2\|G\|$	$3\|G\| + 1\|G_T\|$
[14]	$3\|G\| + 1\|G_T\|$	$2\|G\| + 2\|G_T\|$
Ours	$2\|G\|$	$1\|G\| + 1\|G_T\|$

terms of the above operations. Reference [11] proposed a blockchain-based data trade scheme. Reference [12] proposed a data sharing scheme for the scenario of multiple DCs, in which the PRE and smart contracts were integrated to achieve the privacy-preserving share of medical data. References [13, 14] proposed identity-based PRE approaches to achieve secure data sharing for cloud-assisted systems. We compare our scheme with these schemes [11–14] as we all

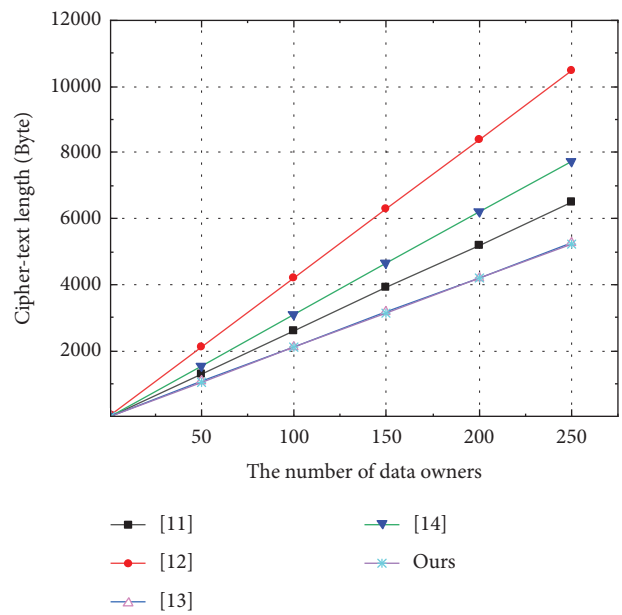


Figure 5: Communication overhead of ciphertext.

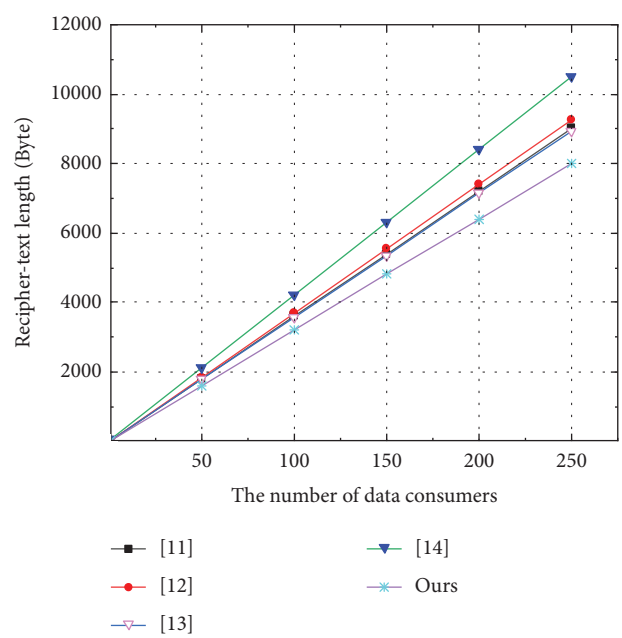


Figure 6: Communication overhead of the reencrypted ciphertext.

utilized the PRE and blockchain and achieved secure data sharing.

Table 3 shows the comparison of operations Par, Exp, and Sm among our scheme and schemes [11–14]. We can see that our scheme needs the fewest Par, Exp, and Sm operations. Figure 4 shows the total time of these schemes, from which we can see that the computation time of data release in our protocol is 14.7 ms, which is much smaller than 23.9, 29.4, 22.3, and 22.9 ms of [11–14]. The computation time of data retrieval in our protocol is 23.2 ms, which is also smaller than 44.6, 63, 42.8, and 66.2 ms of [11–14].

6.2.2. Communication Overhead. To evaluate the communication overhead, we denote the sizes of a scalar value in $\mathbb{Z}_p$, the group elements in $\mathbb{G}$, and in $\mathbb{G}_T$ by $|Z_p|, |G|$, and $|G_T|$, respectively. We choose SHA-256 as the hash function of h. The symmetric encryption algorithm is AES-256. The signature used to sign a transaction of blockchain is ecdsa-with-SHA256. Based on these, Table 4 compares the communication costs in [11–14] in terms of encryption overhead and re-encryption overhead. Figure 5 and Figure 6 show the comparison results, from which we can see that the ciphertext length of our protocol and that of [13] are identical, which are shorter than those in [11, 12, 14], and the length of the re-encryption ciphertext of our scheme is shorter than those in [11–14]. Considering that our scheme has a significant advantage in computational time, our scheme is more efficient in the aspect of both computational overhead and communication overhead.

7. Conclusion

This paper proposes a flexible and secure data sharing method for data-driven systems. In order to ensure confidentiality and reliability, data are encrypted and then stored off-chain. In contrast, the relevant information, such as digest, is stored on-chain, and data can be efficiently shared with authorized users with the help of a CS. The smart contract is employed to control data access such that the key delegation can be automatically executed and the DO is not required to be online all the time. In order to enable security and privacy, the smart contract is executed in the TEEs. Besides, all interactions, data delegations, and other operations are recorded in the blockchain and can be checked at any time, which realizes the efficient monitoring and auditing of the data. We proved that the security properties, such as confidentiality, anonymity, and verifiable integrity, are ensured during the whole data sharing process. We also simulated the proposed scheme, and the results show that our scheme has a better performance than the related works.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the Presidential Foundation of CAEP (Grant No. CX20220001), the Defense Industrial Technology Development Program (JCKY2019602B013), and the Natural Science Foundation (U19A2066).

References

[1] Y. Liu, W. Guo, C.-I. Fan, L. Chang, and C. Cheng, "A practical privacy-preserving data aggregation (3pda) scheme for smart grid," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 3, pp. 1767–1774, 2019.

[2] H. Shafagh, L. Burkhalter, A. Hithnawi, and D. Simon, "Towards blockchain-based auditable storage and sharing of iot data," in *Proceedings of the 2017 on cloud computing security workshop*, pp. 45–50, Dallas TX USA, November 2017.

[3] B. Matt, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," in *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 127–144, Springer, Trondheim, Norway, June 1998.

[4] C. Ran and S. Hohenberger, "Chosen-ciphertext secure proxy re-encryption," in *Proceedings of the 14th ACM conference on Computer and communications security*, pp. 185–194, Alexandria, VA, USA, November 2007.

[5] J. Weng, R. H. Deng, X. Ding, C.-K. Chu, and J. Lai, "Conditional proxy re-encryption secure against chosen-ciphertext attack," in *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, pp. 322–332, Sydney Australia, March 2009.

[6] J. Lai, R. H. Deng, C. Guan, and J. Weng, "Attribute-based encryption with verifiable outsourced decryption," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 8, pp. 1343–1354, 2013.

[7] Q. Liu, G. Wang, and J. Wu, "Time-based proxy re-encryption scheme for secure data sharing in a cloud environment," *Information Sciences*, vol. 258, pp. 355–370, 2014.

[8] L. Xu, X. Wu, and X. Zhang, "Cl-pre: a certificateless proxy re-encryption scheme for secure data sharing with public cloud," in *Proceedings of the 7th ACM symposium on information, computer and communications security*, pp. 87-88, Seoul, Republic of Korea, May 2012.

[9] B. Huang, Z. Liu, J. Chen, A. Liu, Q. Liu, and Q. He, "Behavior pattern clustering in blockchain networks," *Multimedia Tools and Applications*, vol. 76, no. 19, Article ID 20099, 2017.

[10] M. Zhang, C. Chen, T. Wo, T. Xie, M. Z. A. Bhuiyan, and X. Lin, "Safedrive: online driving anomaly detection from large-scale vehicle data," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 4, pp. 2087–2096, 2017.

[11] Y. J. Galteland and S. Wu, *Blockchain-based privacy-preserving fair data trading protocol*, Cryptology ePrint Archive, 2021, https://eprint.iacr.org/2021/1321.

[12] H. Huang, P. Zhu, F. Xiao, X. Sun, and Q. Huang, "A blockchain-based scheme for privacy-preserving and secure sharing of medical data," *Computers & Security*, vol. 99, Article ID 102010, 2020.

[13] K. O.-B. O. Agyekum, Q. Xia, E. B. Sifah, C. N. A. Cobblah, H. Xia, and J. Gao, "A proxy re-encryption approach to secure data sharing in the internet of things based on blockchain," *IEEE Systems Journal*, vol. 16, no. 1, pp. 1685–1696, 2022.

[14] J. He, Z. Dong, R. Guo, Y. Chen, K. Li, and X. Tao, "Efficient identity-based proxy re-encryption scheme in blockchain-assisted decentralized storage system," *International Journal on Network Security*, vol. 23, no. 5, pp. 776–790, 2021.

[15] R. Li, T. Song, Bo Mei, H. Li, X. Cheng, and L. Sun, "Blockchain for large-scale internet of things data storage and protection," *IEEE Transactions on Services Computing*, vol. 12, no. 5, pp. 762–771, 2019.

[16] T. McConaghy, R. Marques, A. Müller et al., *Bigchaindb: A Scalable Blockchain Database*, white paper, BigChainDB, Berlin, Germany, 2016.

[17] T. Li, H. Wang, D. He, and J. Yu, "Blockchain-based privacy-preserving and rewarding private data sharing for iot," *IEEE*

Internet of Things Journal, vol. 9, no. 16, Article ID 15138, 2022.

[18] Y. Wang, Z. Su, N. Zhang et al., "Spds: a secure and auditable private data sharing scheme for smart grid based on blockchain," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 11, pp. 7688–7699, 2021.

[19] H. Lei, Y. Yan, Z. Bao, Q. Wang, Y. Zhang, and W. Shi, "Sdsbt: a secure multi-party data sharing platform based on blockchain and tee," in *Proceedings of the International Symposium on Cyberspace Safety and Security*, pp. 184–196, Springer, Haikou, China, December 2020.

[20] S. Nakamoto and A. Bitcoin, "A peer-to-peer electronic cash system," 2008, https://bitcoin.org/bitcoin.pdf.

[21] C. Linnhoff-Popien, R. Schneider, and M. Zaddach, *Digital Marketplaces Unleashed*, Springer, Singapore, 2018.

[22] V. Karande, E. Bauman, Z. Lin, and L. Khan, "Sgx-log: securing system logs with sgx," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pp. 19–30, Abu Dhabi, UAE, April 2017.

[23] N. Santos, H. Raj, S. Saroiu, and A. Wolman, "Using arm trustzone to build a trusted language runtime for mobile applications," in *Proceedings of the 19th international conference on Architectural support for programming languages and operating systems*, pp. 67–80, Salt Lake City, UT, USA, March 2014.

[24] C. Garlati and S. Pinto, "A clean slate approach to linux security risc-v enclaves," in *Proceedings of the Embedded World Conference*, p. 5, Nuremberg, Germany, February 2020.

[25] M. Bowman, A. Miele, M. Steiner, and V. Bruno, "Private Data Objects: An Overview," 2018, https://arxiv.org/abs/1807.05686.

[26] B. V. B. M. Andrea Miele and A. Adesokan, "Private-data-objects," https://github.com/hyperledger-labs/private-data-objects.

[27] R. Cheng, F. Zhang, J. Kos et al., "Ekiden: a platform for confidentiality-preserving, trustworthy, and performant smart contracts," in *Proceedings of the 2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 185–200, IEEE, Stockholm, Sweden, June 2019.

[28] A. Prashanth Joshi, M. Han, and Y. Wang, "A survey on security and privacy issues of blockchain technology," *Mathematical Foundations of Computing*, vol. 1, no. 2, pp. 121–147, 2018.

[29] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612-613, 1979.

[30] D. Schultz, B. Liskov, and M. Liskov, "Mpss: mobile proactive secret sharing," *ACM Transactions on Information and System Security*, vol. 13, no. 4, pp. 1–32, 2010.

[31] M. Scott, K. McCusker, A. Budroni, and S. Andreoli, "The Miracl Core Cryptographic Library," 2019, https://github.com/miracl/core.