Journal of Computer Networks and Communications

Advances in Machine Learning for Cybersecurity

Lead Guest Editor: Mazdak Zamani
Guest Editors: Saman S. Chaeikar and Arash H. Lashkari

Advances in Machine Learning for Cybersecurity

Journal of Computer Networks and Communications

Advances in Machine Learning for Cybersecurity

Lead Guest Editor: Mazdak Zamani
Guest Editors: Saman S. Chaeikar and Arash H. Lashkari

Academic Editors

Contents

Hindawi
Journal of Computer Networks and Communications
Volume 2020, Article ID 6576725, 1 page
https://doi.org/10.1155/2020/6576725

Corrigendum

Corrigendum to "Multistage System-Based Machine Learning Techniques for Intrusion Detection in WiFi Network"

Vu Viet Thang[1,2] **and F. F. Pashchenko**[3,4]

[1]*Moscow Institute of Physics and Technology (State University), Moscow, Russia*
[2]*Faculty of Information Technology, Hanoi University of Industry, Hanoi, Vietnam*
[3]*The Department of Information and Communication Technologies, MIPT (State University), Moscow, Russia*
[4]*Trapeznikov Institute of Control Sciences, Russian Academy of Sciences, Moscow, Russia*

Correspondence should be addressed to Vu Viet Thang; vv.tkhang@phystech.edu

Received 9 August 2020; Accepted 9 August 2020; Published 8 September 2020

In the article titled "Multistage System-Based Machine Learning Techniques for Intrusion Detection in WiFi Network" [1], there was a missing affiliation for the second author. The corrected authors' list and affiliations are shown above.

References

[1] V. V. Thang and F. F. Pashchenko, "Multistage system-based machine learning Techniques for intrusion detection in WiFi Network," *Journal of Computer Networks and Communications*, vol. 2019, Article ID 4708201, 13 pages, 2019.

Hindawi
Journal of Computer Networks and Communications
Volume 2019, Article ID 5124364, 10 pages
https://doi.org/10.1155/2019/5124364

Research Article

An Optimized Approach for Secure Data Transmission Using Spread Spectrum Audio Steganography, Chaos Theory, and Social Impact Theory Optimizer

Rohit Tanwar,[1] Kulvinder Singh,[2] Mazdak Zamani,[3] Amit Verma,[1] and Prashant Kumar[1]

[1]*University of Petroleum and Energy Studies, Dehradun, India*
[2]*UIET Kurukshetra University, Kurukshetra, India*
[3]*Felician University, Lodi, New Jersey, USA*

Correspondence should be addressed to Rohit Tanwar; rohit.tanwar.cse@gmail.com

Received 9 April 2019; Revised 10 July 2019; Accepted 29 July 2019; Published 8 September 2019

Academic Editor: Liansheng Tan

Being easy to understand and simple to implement, substitution technique of performing steganography has gain wide popularity among users as well as attackers. Steganography is categorized into different types based on the carrier file being used for embedding data. The audio file is focused on hiding data in this paper. Human has associated an acute degree of sensitivity to additive random noise. An individual is able to detect noise in an audio file as low as one part in 10 million. Given this limitation, it seems that concealing information within audio files would be a pointless exercise. Human auditory system (HAS) experiences an interesting behavior known as masking effect, which says that the threshold of hearing of one type of sound is affected by the presence of another type of sound. Because of this property, it is possible to hide some data inside an audio file without being noticed. In this paper, the research problem for optimizing the audio steganography technique is laid down. In the end, a methodology is proposed that effectively resolves the stated research problem and finally the implementation results are analyzed to ensure the effectiveness of the given solution.

1. Introduction

Currently, audio steganography is limited to providing solutions related to copyright and assurance of the integrity of content [1]. It is possible to expand its applications to incorporate the embedding of covert communications as well. Various audio steganography strategies are developed and utilized for embedding knowledge in audio files. These strategies embrace low-bit encoding (LSB), polarity inversion, echo activity, part committal to writing, cepstral activity, sensory activity masking, and spread spectrum [2, 3]. Low-bit encoding (LSB) is the most famous technique, but it suffers from some severe limitations. Out of other techniques of communication, spread spectrum is receiving increasing attention. Spread spectrum technology forms the premise of spread spectrum steganography [4–7]. Because of the higher data hiding capacity of the low-bit encoding substitution technique, its alternate is expected to have similar capacity while maintaining better robustness and imperceptibility. Therefore, there is a requirement for incorporating optimization technique with a steganography method so that the required expectations can be fulfilled effectively [8–12].

2. Motivation and Scope for Optimization

The effectiveness of a steganography technique is analyzed on the parameters, namely, capacity, imperceptibility, and robustness. In most of the existing techniques, adherence with one of these parameters is usually achieved at the cost of comprising with other parameters. Low-bit encoding or LSB technique is the most commonly used technique [13–19].

However, the inherent limitations of the LSB method that are listed below inspire to look for an effective and optimized alternate approach [20–24]:

(i) LSB method is prone to intentional attacks as data are embedded in LSBs only

(ii) Unintentional attacks like noise disturbances are the cause of data loss

The literature relevant to audio steganography techniques was studied [25–49], and the observations are summarized as a comparison of the techniques on the basic parameters, like robustness, imperceptibility, capacity, and complexity, as shown in Table 1. The analysis says that it is difficult for an existing technique to achieve the satisfactory value of all the comparison parameters. A method that yields high capacity does not have adequate robustness and imperceptibility. Similarly, if robustness was there, the capacity is compromised. Therefore, an audio steganography technique needs to be optimized so that the acceptable values of evaluation parameters are achieved.

Table 1: Comparison of audio steganography techniques.

Techniques	Parameters			
	Imperceptibility	Capacity	Robustness	Complexity
LSB	Medium	High	Low	Low
Parity coding	Medium	Medium	Low	Low
Phase coding	High	Low	High	Medium
Spread spectrum	Low	Medium	High	High
Echo hiding	Low	Medium	Medium	Medium
Tone insertion	Low	Low	Medium	High

3. Problem Formulation

The objective is to design an optimized method that should be having a capacity similar to the LSB technique, but robustness should be high, unlike the LSB method. However, there is always a tradeoff in achieving capacity along with keeping the high value of robustness. Thus, the research problem is stated using the following points:

(i) An audio steganography technique is desired that should have the capacity comparable to the LSB technique but maintaining a high value of robustness and imperceptibility.

(ii) Most of the techniques of implementing audio steganography that possesses satisfactory robustness have less hiding capacity. Therefore, the selected technique is customized to meet the requirements.

(iii) As a consequence of the customization for high capacity, more distortion will get induced in the cover file after embedding. To minimize that distortion, an optimization algorithm is used in coordination with the embedding algorithm.

(iv) The sequence of embedding message bits in the audio file should not be obvious. The embedding pattern is expected to be as less predictable as possible to make the technique more secure.

4. Proposed Methodology

The proposed methodology resolves the research problem in an efficient manner. A coordinated approach using spread spectrum steganography, SITO, and chaos theory works in a way that the objectives of the research problem are satisfied. The underlying representation of the proposed method is shown in Figure 1.

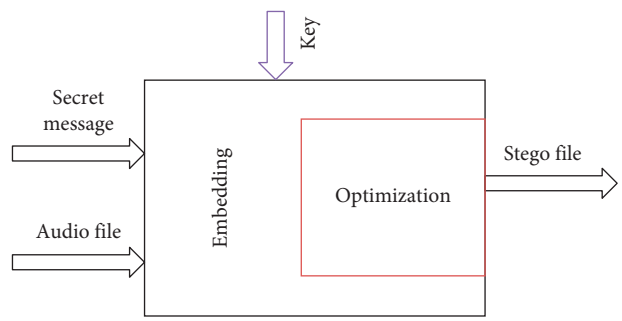


Figure 1: Proposed solution representation.

The proposed method comprised the following basic modules:

(i) Chaotic system: chaotic map is used to provide required randomness as the pseudorandom numbers generated through some random library function result in a repetition of some pattern to be used for substitution. The cross correlation between two consecutive numbers generated through chaotic map approaches to zero. The key is obtained using logistic maps of chaos theory.

(i) Spread spectrum: spread spectrum technique of implementing audio steganography is used as one of the prime components of the proposed solution. Because of its inherent robustness, difficult to intercept, and hard to interference, the technique is the first choice in solving the given research problem.

(ii) Social impact theory optimization: to minimize the distortion induced in audio samples because of embedding, social impact theory optimization (SITO) is used. In recent researches, SITO has outperformed genetic algorithm (GA) and particle swarm optimization (PSO) in achieving optimum results. The algorithm evolves iteration by iteration based on some objective functions and guarantees a nearly optimal solution at the end. The stopping criterion for the algorithm in the current research is the number of iterations.

4.1. Proposed Algorithm. The methodology starts with a capacity check operation to verify the capability of the chosen cover audio file to embed the secret message of given

```
Step 1: procedure Embed_Optimize(aud_file, img_file)
Step 2: if(capacity_check(aud_file, img_file) = TRUE) then
Step 3:   Temp_aud⟵transform(aud_file)
Step 4:     Temp_img⟵Binary(img_file)
Step 5:     seed⟵random()
Step 6:     key⟵seed
Step 7:     count⟵0
Step 8:     while no_of_iteration ≤ 100 do
Step 9:       if count < length(Temp_img) then
Step 10:          key⟵SITO_OPT(key)
Step 11:        hide(Temp_aud, key, Temp_img)
Step 12:          count⟵update(count)
Step 13:        else
Step 14:          Write"Message got embedded successfully".
Step 15:     repeat
Step 16:     return Temp_aud
Step 17:     else
Step 18:     exit()
Step 19:     return
```

Algorithm 1: Optim_Audio, input: Audio file, image file, output: stego file.

size successfully. After that, the cover audio file is transformed into its equivalent spread spectrum. Simultaneously, the secret message is converted into its binary equivalent. The required random sequence that acts as a key to guide the spreading of message bits over the entire spectrum of the cover file is obtained using the logistic map of chaotic theory. The embedding step is repeated until the complete message is hidden inside the cover file. The amount of data that is embedded per sample of the audio file is comparatively more so that capacity is increased. The initial values are calculated for evaluation parameters as well as the objective function. The proposed algorithm is given in Algorithm 1.

In order to optimize the value of the objective function, SITO is used. The values of evaluation parameters are improved with the iterations of the optimization algorithm. The SITO terminates when an acceptable value of evaluation parameters is achieved or the number of iterations is over. The flow chart given in Figure 2 describes the proposed methodology.

4.1.1. Objective Function. The objective is to minimize MSE and maximize PSNR and SSIM. The aggregate function used here is a minimizing function given as

$$f\colon \min\left[\frac{\text{MSE}}{\text{PSNR}\times\text{SSIM}}\right], \tag{1}$$

where MSE stands for mean square error, PSNR stands for peak signal to noise ratio, and SSIM stands for structural similarity index.

5. Implementation

MATLAB is a multiparadigm environment developed for numerical computing primarily. With time, it got equipped with a list of toolboxes developed for specific needs. Initially, the command window was the only way to interact for executing files. Later, an additional feature of graphical interface benefitted the users. Because of its broad applicability, popularity, and effectiveness, the proposed algorithm is implemented using MATLAB. The open-source libraries are used for implementing chaos theory and SITO. To compare and analyze the effectiveness of the proposed algorithm, the comparison graphs are generated for PSNR, MSE, and SSIM. MATLAB 2014 is used as the simulation environment. The details regarding secret message embedded, cover audio file, and libraries used are given in the following:

(i) Secret message: (i) JPG image of size 8.78 KB and dimension 244×250; (ii) PNG image of size 38.6 KB and dimension 512×512

(ii) Cover audio: MP4 file of size 5.07 MB and length 3 min 21 sec, bit rate 206 kbps, channels 2, and audio sample rate 44.100 kHz

(iii) CODO library: library used for random number generator using chaos theory

(iv) SITO library: library used for optimizer using social impact theory

6. Result Analysis

The effectiveness of the proposed solution of the formulated problem is evaluated through the following methods:

(i) Histogram analysis of original message and extracted message

(ii) Frequency spectrum analysis of cover audio and stego audio

(iii) Peak signal to noise ratio (PSNR)

(iv) Mean square error (MSE)

(v) Structural similarity index (SSIM)

(vi) Visual inspection of the original message and extracted message

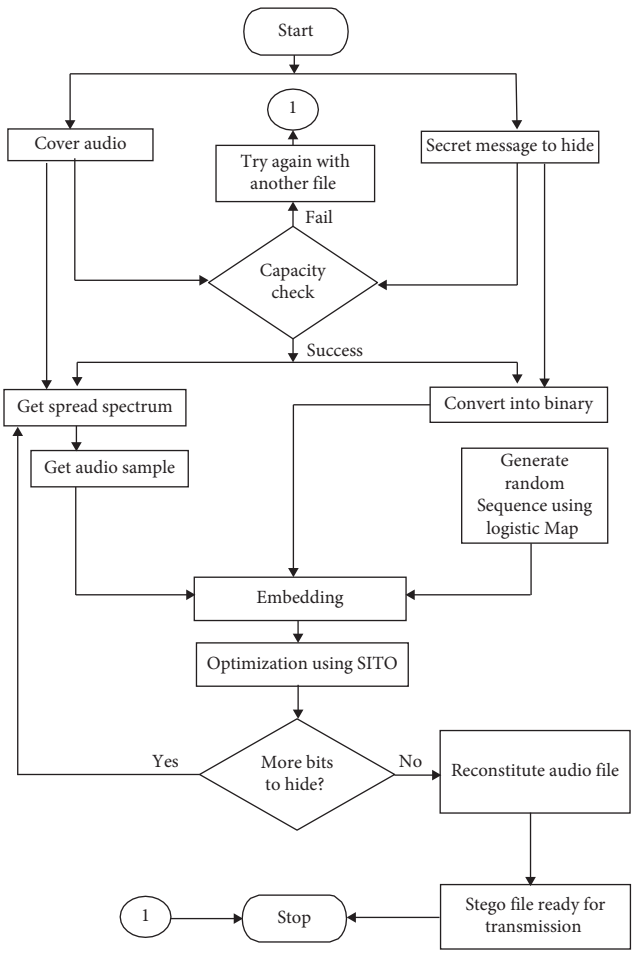


Figure 2: Flow chart of the proposed methodology.

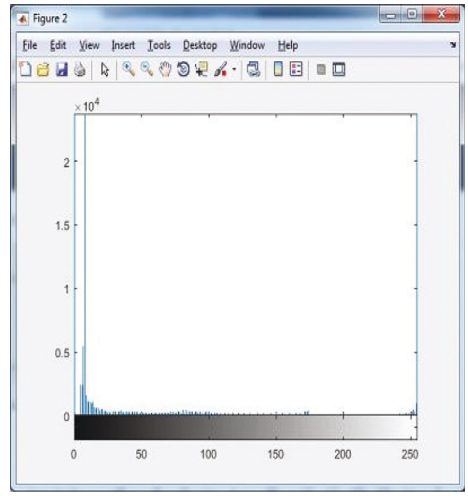
(vii) Robustness measurement using correlation coefficient

(viii) Comparison of variation of PSNR, MSE, and SSIM with some existing techniques

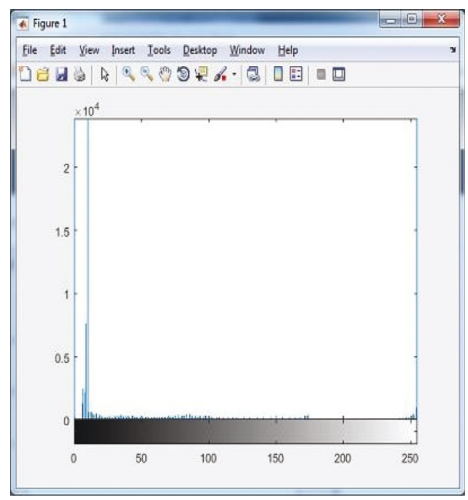
6.1. Histogram of Original Image and Extracted Image. An image is taken as secret data to be embedded in audio. Histogram representation of an image is a popular way of analyzing images.

Either the histograms can be taken of the single channel out of RGB or it can be taken of RGB in aggregate. The given histograms are obtained for blue channels of the image before embedding and extracted image. Similarly, analysis of histograms of red and green color can be done. The histogram comparisons in Figure 3 say that a histogram of the message extracted after embedding using the proposed method is closer to the histogram of the original image.

6.2. Frequency Spectrum of Cover Audio and Stego Audio. Obtaining the frequency spectrum is a universal and straightforward technique to analyze an electric signal. Figure 4 represents the snapshots of the frequency spectrum of the audio signal before and after embedding. Frequency



(a)



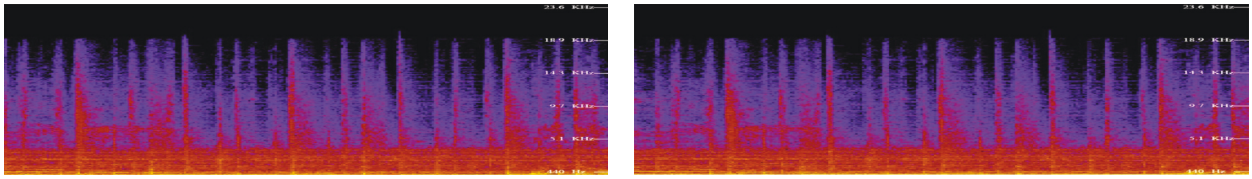
(b)

Figure 3: Histograms of (a) original message and (b) extracted message.

spectrums of the stego files after embedding of the secret message using the proposed algorithm and prior to embedding are obtained, and the snapshots are captured for the same interval so as to fit them on page size, as shown in Figure 4. The frequency spectrum of the stego audio is very similar to that of cover audio. Since the audio file is of long duration, the snapshots are captured for a specific time interval, as shown in Figure 4.

6.3. Analysis Using PSNR, MSE, and SSIM. PSNR, MSE, and SSIM have been widely accepted as a measure of quality. To verify the quality of the proposed method, the variation of PSNR, MSE, and SSIM values is obtained. PSNR and MSE are to calculate absolute error, but SSIM gives a measure of error in the structure.

It also looks for intensity, brightness, and other parameters that are related to the structure of an image. The unit of measuring PSNR is dB, and the acceptable value is greater than 45 dB, whereas MSE has the unit square of the unit of the quantity being measured. The desired value of



(a) (b)

Figure 4: Frequency spectrum of cover audio (a) and stego audio (b).

MSE should be approaching to "0." The value of SSIM lies between "−1" and "+1." The desired value of SSIM should be approaching to "+1." "0" value indicates no similarity while value "+1" indicates that both images are identical. The results show that the improvement is significant and it would be increasing with the increase in some iteration of the proposed method. Figure 5 shows the PSNR, MSE, and SSIM variation as per the increasing iterations that reflect the significant improvement gained by using the proposed work.

6.4. Visual Inspection of Original Message and Extracted Message. The secret messages before embedding and the hidden messages extracted from stego file are compared and analyzed for any visually noticeable differences in Figure 6. The authentic recipient can easily recognize and evaluate secret message after extraction from stego audio file. The use of some reconstruction algorithm at the receiver's end may result in an identical image available for processing after extraction.

6.5. Robustness Measurement. Robustness is that property of the watermarked data that make its presence silent even after attacked with general signal processing attacks [50, 51]. The most common parameter for measuring robustness is the correlation coefficient. The higher value of correlation coefficient of original image and watermarked image is a measure of more robustness. The audio cover file with embedded watermarked is attacked with compression. The correlation coefficient was calculated after the attack for both the secret messages. The average value of correlation coefficient (Corr_{avg}) comes out to be 0.85.

6.6. Comparison with Existing Techniques. It is essential to examine and compare the efficacy of the proposed research with the preexisting techniques in a similar domain. The proposed work of optimizing audio steganography using social impact theory optimization is compared with some of the existing methods on different parameters like PSNR, MSE, and SSIM. The comparison is made using the following: (i) the traditional LSB method, (ii) another optimized steganography technique where algorithm used for optimization is GA, and (iii) the implementation work done by Chen and Huang [1].

6.6.1. Comparison of PSNR Variation. PSNR variation is compared in Figure 7, with the LSB method, with GA, and with the research work of Chen and Huang [1]. It is observed

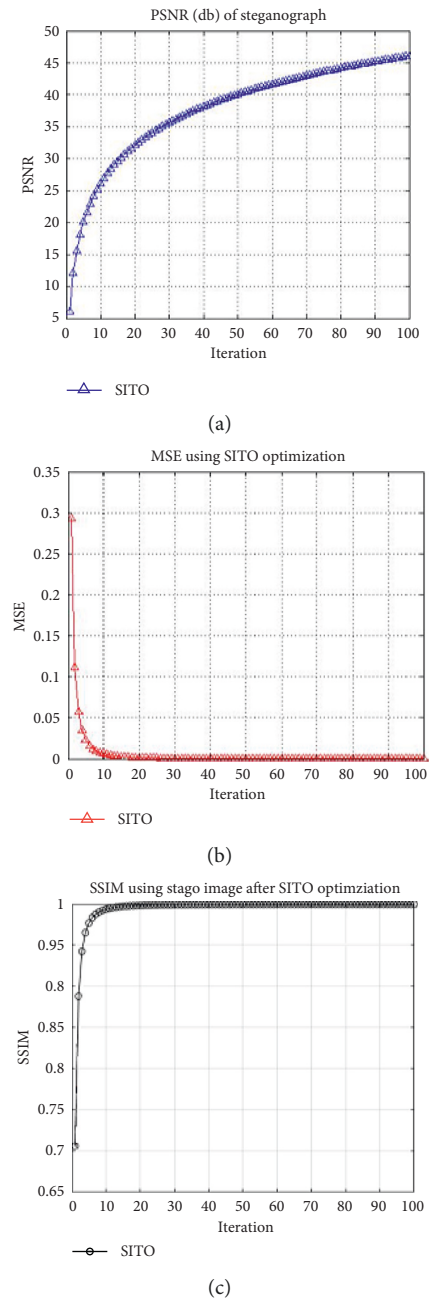
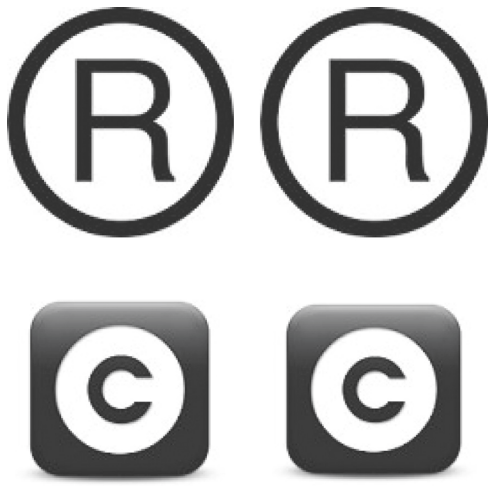


Figure 5: (a) PSNR, (b) MSE, and (c) SSIM variation.

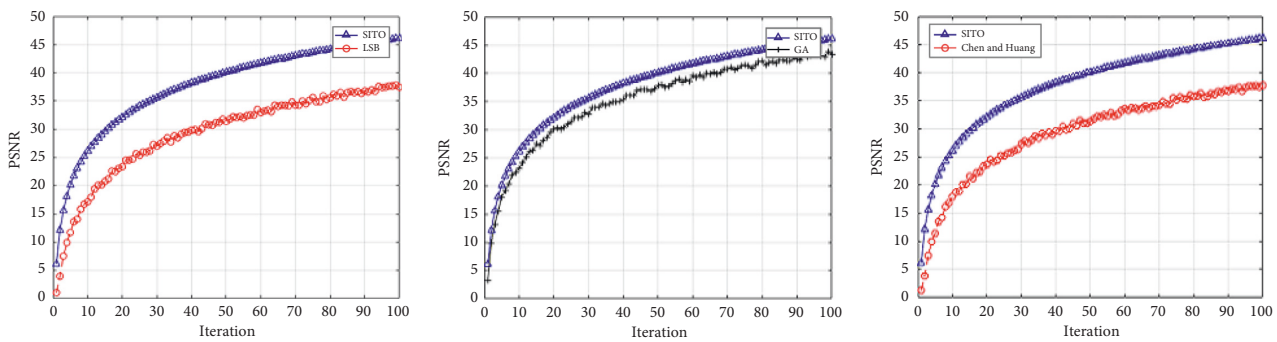
from these graphs that the PSNR achieved is satisfactory while SITO is used for optimization in the proposed work.

However, the graph diverges further when the number of iterations increases. The proposed work outperforms the



(a) (b)

Figure 6: (a) Original message and (b) extracted message.



(a) (b) (c)

Figure 7: Comparison of PSNR variation with (a) LSB, (b) GA, and (c) the method of Chen and Huang [1].

LSB technique and the method of Chen and Huang [1] in case of PSNR variation. The performance greatly improves when the algorithm iterates SITO for more number of times. Figure 7 also indicates that the proposed method is far better than the LSB technique in the attainment of PSNR. Moreover, improvement in the PSNR value was observed even after few initial iterations. Similarly, GA behaves like as proposed work in initial few iterations, but after that, the variation of PSNR with respect to increase in the number of iterations starts decreasing. However, PSNR values attained by using proposed work and GA differ in the range of "0" dB to "5" dB maximum. It can be seen from the figure that the research work done by Chen and Huang [1] behave similar to the least significant bit method in the attainment of PSNR. Proposed works attain considerably greater values of PSNR as compared to the research work of Chen and Huang [1] with respect to the increase in the number of iterations.

The comparison of PSNR variation also shows that LSB technique behaved in the least effective way, while GA was comparable to the proposed method and the third one behaved in a moderate way.

6.6.2. Comparison of MSE Variation. It was observed while comparing the proposed work with the LSB method, GA, and the method of Chen and Huang [1] that GA and LSB yielded almost similar results; however, the proposed work outperformed other three techniques in this category of comparison of MSE variation.

Starting from a very high value of MSE, the proposed algorithm abruptly comes down and sooner attains the acceptance value. One more fact observed from the graphs given in Figure 8 is a little variation of MSE in all the three techniques except SITO.

6.6.3. Comparison of SSIM Variation. Structural similarity index (SSIM) is a way of measuring degradation in the quality of an image because of some processing task.

The acceptable value of SSIM should be close to one. From Figure 9, it is clear that the proposed method achieves a better value of SSIM as compared to LSB, GA, and research of Chen and Huang [1]. However, GA performs better than the other two techniques.

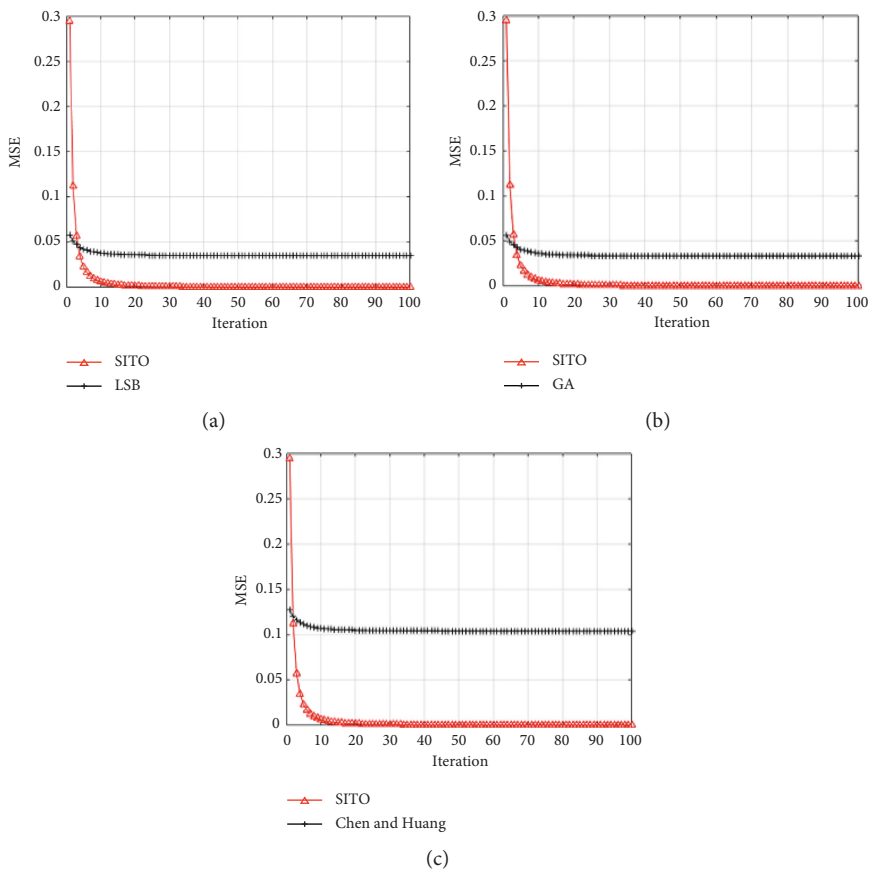


Figure 8: Comparison of MSE variation with (a) LSB, (b) GA, and (c) the method of Chen and Huang [1].

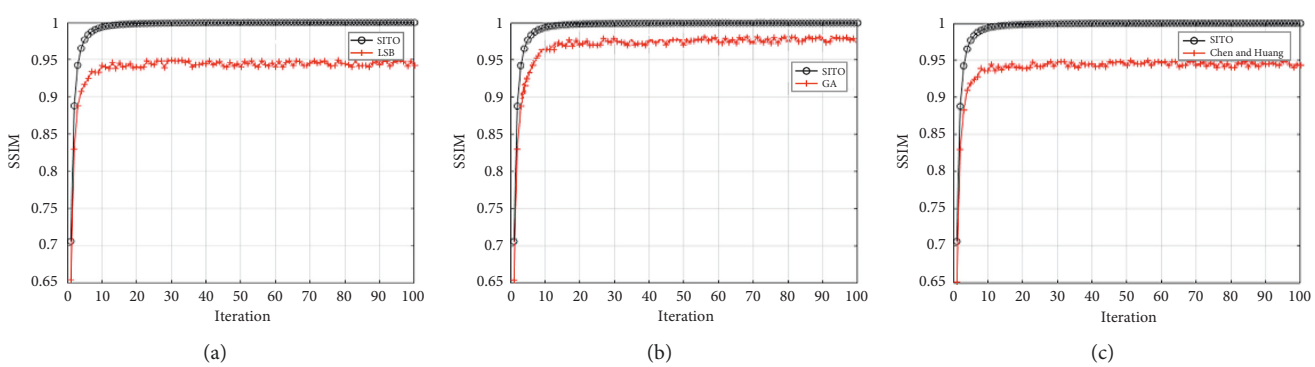


Figure 9: Comparison of SSIM variation with (a) LSB, (b) GA, and (c) the method of Chen and Huang [1].

6.6.4. Comparison with the Works of Su et al. [50] and Lei et al. [51]. The attainment of imperceptibility and robustness depends largely on scaling parameter chosen for embedding. Most of the audio steganography implementations use static value of the scaling parameter for simplicity. In the work done by Su et al. [50], an optimal method for the selection of scaling parameter is described that needs comparatively less computation. The optimal selection of the value of scaling parameter results in higher SNR value and larger robustness (correlation coefficient up to 0.99). On the other hand, Lei et al. [51] suggested a customized objective function and used self-adaptive particle swarm optimization technique along with quaternion wavelet transform. In their approach, a higher value of coefficient correlation (0.99, best value) is achieved. It can be said that the optimal choice of scaling parameter in [50] and use of modified PSO in [51] outperformed the current work.

(1) Comparison of PSNR, MSE, and SSIM Value Attainment. Figures 10–12 compare the PSNR, MSE, and SSIM values achieved in each of the techniques discussed above. It is clearly deduced from the respective figures that the proposed methodology attains satisfactory and better values of PSNR, MSE, and SSIM comparatively. Genetic algorithm also performed well in attaining the evaluation parameters, but still it could not overtake the proposed work.

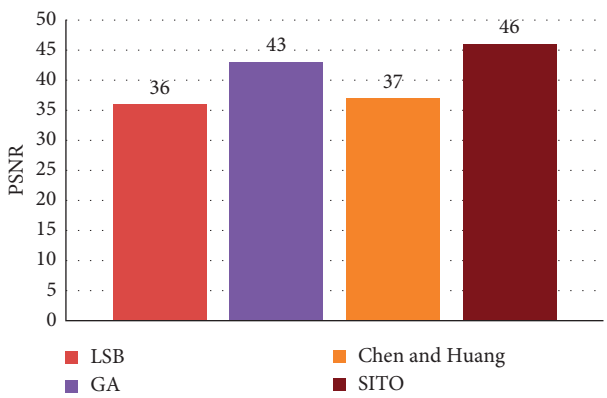


Figure 10: PSNR attainment.

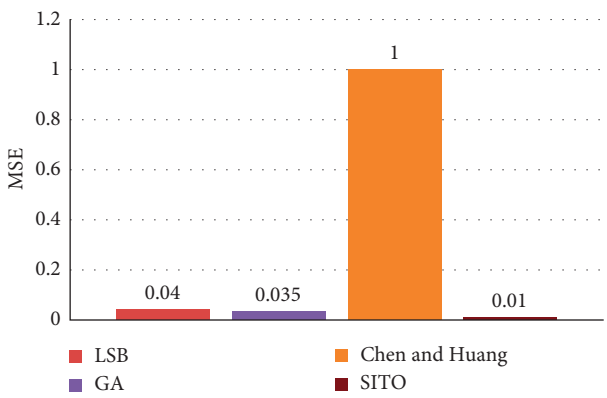


Figure 11: MSE attainment.

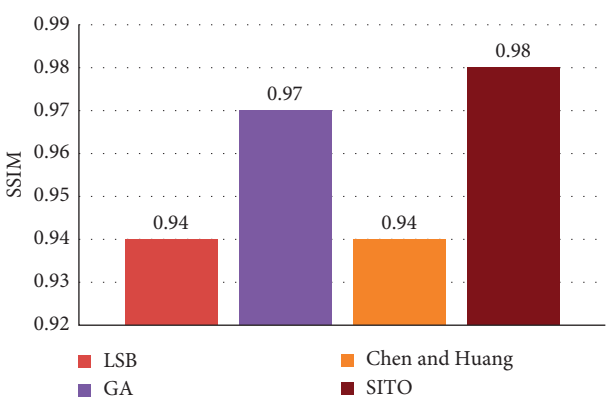


Figure 12: SSIM attainment.

7. Conclusion

The problem of finding an audio steganography technique with acceptable values of capacity, robustness, and imperceptibility is resolved using the proposed methodology. The characteristics of spread spectrum that makes it secure against interception and interference provide the required robustness. The prime security of steganography lies in difficulty to know the hiding pattern used for embedding. Thus, the security is ensured by making the embedding pattern truly random by utilizing chaos theory using logistic maps. Capacity is increased by spreading a greater number of bits over the entire spectrum of a sample. This enhancement would increase the distortion too. The SITO maintains the distortion at an optimum level and optimizes the audio steganography technique to achieve the satisfactory value of the objective function. It is observed from the analysis of different graphs that significant improvement has been achieved by using social impact theory optimizer. The performance of the proposed research work improves further with the increase in the number of iterations of SITO execution. Various quality measures (PSNR, MSE, and SSIM) achieved values up to satisfaction. In some cases, GA performed somewhere close to SITO but the performance gain observed by using SITO as an optimization algorithm is more significant. The proposed methodology successfully achieves the research objectives of optimizing an audio steganography algorithm in such a way that each parameter out of robustness, capacity, and imperceptibility is attained to the satisfaction.

Data Availability

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

Conflicts of Interest

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

References

[1] S.-T. Chen and H.-N. Huang, "Optimization-based audio watermarking with integrated quantization embedding," *Multimedia Tools and Applications*, vol. 75, no. 8, pp. 4735–4751, 2016.

[2] S. K. Pal, P. K. Saxena, and S. K. Muttoo, "The future of audio steganography," in *Proceedings of the Pacific Rim Workshop on Digital Steganography 2002 (STEG'02)*, Kitakyushu, Japan, July 2002.

[3] D. Sellars, "An introduction to steganography," February 2003, http://www.totse.com/en/privacy/encryption/163947.html.

[4] S. S. Chaeikar, M. Zamani, A. B. A. Manaf, and A. M. Zeki, "PSW statistical LSB image steganalysis," *Multimedia Tools and Applications*, vol. 77, no. 1, pp. 805–835, 2018.

[5] F. Arab and M. Zamani, "VW16F: a robust video watermarking using simulated block based spatial domain technique," in *Proceedings of the International Conference on Security and Management*, Las Vegas, USA, July 2018.

[6] F. Arab and M. Zamani, "VW16E: a robust video watermarking technique using simulated blocks," *Intelligent Systems Reference Library*, vol. 115, pp. 193–221, 2017.

[7] F. Arab, S. M. Abdullah, S. Z. M. Hashim, A. A. Manaf, and M. Zamani, "A robust video watermarking technique for the tamper detection of surveillance systems," *Multimedia Tools and Applications*, vol. 75, no. 18, pp. 10855–10885, 2016.

[8] T. K. Araghi, A. B. A. Manaf, M. Zamani, and S. K. Araghi, "A survey on digital image watermarking techniques in spatial and transform domains," *International Journal of Advances in Image Processing Techniques*, vol. 3, no. 1, pp. 6–10, 2016.

[9] M. Zamani and A. B. A. Manaf, "Genetic algorithm for fragile audio watermarking," *Telecommunication Systems*, vol. 59, no. 3, pp. 291–304, 2015.

[10] M. Zamani, A. Abdul Manaf, and R. Ahmad, "Current problems of substitution technique of audio steganography," in *Proceedings of the International Conference on Artificial Intelligence and Pattern Recognition*, Orlando, USA, July 2009.

[11] M. Zamani, A. Abdul Manaf, and R. Ahmad, "Genetic algorithm as an approach to resolve the problems of substitution techniques of audio steganography," in *Proceedings of the International Conference on Genetic and Evolutionary Methods*, Las Vegas, USA, July 2009.

[12] M. Zamani, A. Abdul Manaf, and R. Ahmad, "Knots of substitution techniques of audio steganography," in *Proceedings of the International Conference on Telecom Technology and Applications*, Manila, Philippines, June 2009.

[13] M. Zamani, A. Abdul Manaf, R. Ahmad, F. Jaryani, H. Taherdoost, and A. Zeki, "A secure audio steganography approach," in *Proceedings of the 4th International Conference for Internet Technology and Secured Transactions*, London, UK, November 2009.

[14] M. Zamani, A. Abdul Manaf, R. Ahmad, A. Zeki, and S. Abdullah, "A genetic-algorithm-based approach for audio steganography," *World Academy of Science, Engineering and Technology*, vol. 30, pp. 355–358, 2009.

[15] M. Zamani, A. Abdul Manaf, R. Ahmad, A. Zeki, and P. Magalingam, "A novel approach for audio watermarking," in *Proceedings of the Fifth International Conference on Information Assurance and Security*, Xian, China, August 2009.

[16] M. Zamani, R. Ahmad, A. Abdul Manaf, and A. Zeki, "An approach to improve the robustness of substitution techniques of audio steganography," in *Proceedings of the International Conference on Computer Science and Information Technology*, Beijing, China, August 2009.

[17] M. Zamani, H. Taherdoost, A. Abdul Manaf, R. Ahmad, and A. Zeki, "An artificial-intelligence-based approach for audio steganography," *MASAUM Journal of Open Problems in Science and Engineering*, vol. 1, no. 1, pp. 64–68, 2009.

[18] M. Zamani, H. Taherdoost, A. Abdul Manaf, R. Ahmad, and A. Zeki, "Robust audio steganography via genetic algorithm," in *Proceedings of the International Conference on Information & Communication Technologies*, Karachi, Pakistan, August 2009.

[19] S. Abdullah, A. Abdul Manaf, and M. Zamani, "Capacity and quality improvement in reversible image watermarking approach," in *Proceedings of the International Conference on Networked Computing and Advanced Information Management*, Seoul, South Korea, August 2010.

[20] S. Abdullah, A. Abdul Manaf, and M. Zamani, "Recursive reversible image watermarking using enhancement of difference expansion techniques," *Journal of Information Security Research*, vol. 1, no. 2, pp. 64–70, 2010.

[21] M. Zamani and A. Abdul Manaf, "Azizah's method to measure the efficiency of steganography techniques," in *Proceedings of the 2nd International Conference on Information and Multimedia Technology*, Hong Kong, China, December 2010.

[22] M. Zamani and A. Abdul Manaf, "Mazdak's method for PSNR estimation in audio steganography," in *Proceedings of the International Conference on Computer and Computational Intelligence*, Nanning, China, December 2010.

[23] M. Zamani, A. Abdul Manaf, R. Ahmad, F. Jaryani, S. Shojae Chaeikar, and H. Rouhani Zeidanloo, "Genetic audio watermarking," in *Communications in Computer and Information Science*, vol. 70, pp. 514–517, Springer, Berlin, Heidelberg, Germany, 2010.

[24] M. Zamani, A. Abdul Manaf, R. Ahmad et al., "Genetic audio steganography," *International Journal on Recent Trends in Engineering & Technology*, vol. 3, no. 2, pp. 89–91, 2010.

[25] M. Zamani, A. Abdul Manaf, R. Ahmad et al., "A novel approach for genetic audio watermarking," *Journal of Information Assurance and Security*, vol. 5, no. 1, pp. 102–111, 2010.

[26] A. Zeki, A. Abdul Manaf, and M. Zamani, "Bit-plane model: theory and implementation," in *Proceedings of the Engineering Conference*, Kuching, Malaysia, August 2010.

[27] M. Zamani, A. B. A. Manaf, H. R. Zeidanloo, and S. S. Chaeikar, "Genetic substitution-based audio steganography for high capacity applications," *International Journal of Internet Technology and Secured Transactions*, vol. 3, no. 1, pp. 97–110, 2011.

[28] A. M. Zeki, A. A. Manaf, A. A. Ibrahim, and M. Zamani, "A robust watermark embedding in smooth areas," *Research Journal of Information Technology*, vol. 3, no. 2, pp. 123–131, 2011.

[29] A. Jabbar Altaay, S. Bin Sahib, and M. Zamani, "An introduction to image steganography techniques," in *Proceedings of the International Conference on Advanced Computer Science Applications and Technologies*, Kuala Lumpur, Malaysia, November 2012.

[30] M. Zamani, A. Abdul Manaf, and S. Abdullah, *Correlation between PSNR and Size Ratio in Audio Steganography Recent Researches in Communications, Signals and Information Technology*, WSEAS, Greece, 2012.

[31] M. Zamani, A. Abdul Manaf, and S. Abdullah, *Efficient Embedding for Audio Steganography Models and Methods in Applied Sciences*, WSEAS, Greece, 2012.

[32] M. Zamani, A. Abdul Manaf, and S. Abdullah, "An overview on audio steganography techniques," *International Journal of Digital Content Technology and Its Applications*, vol. 6, no. 13, pp. 107–122, 2012.

[33] M. Zamani, A. Abdul Manaf, S. Abdullah, and S. Shojae Chaeikar, *Correlation between PSNR and Bit Per Sample Rate in Audio Steganography Recent Researches in Communications, Signals and Information Technology*, WSEAS, Greece, 2012.

[34] M. Zamani, A. Abdul Manaf, S. Abdullah, and S. Shojae Chaeikar, *Mazdak Technique for PSNR Estimation in Audio Steganography Applied Mechanics and Materials*, Vol. 229, Trans Tech Publications Inc., Zürich, Switzerland, 2012.

[35] M. Zamani, A. Abdul Manaf, and R. Daruis, "Azizah technique for efficiency measurement in steganography," in *Proceedings of the 8th International Conference on Information Science and Digital Content Technology*, Jeju, South Korea, June 2012.

[36] A. Jabbar Altaay, S. Bin Sahib, and M. Zamani, *Correlation Analysis of the Four Photo Themes in Five Layers Communications in Computer and Information Science*, vol. 398, pp. 211–222, Springer, Berlin, Heidelberg, Germany, 2013.

[37] A. Jabbar Altaay, S. Bin Sahib, and M. Zamani, "Determining the threshold of pixel correlativity analysis," *International Journal of Digital Content Technology and Its Applications*, vol. 7, no. 11, pp. 164–172, 2013.

[38] A. Jabbar Altaay, S. Bin Sahib, and M. Zamani, "Edgy photos statistical steganalysis," *Advances in Information Sciences and Service Sciences*, vol. 5, no. 16, pp. 35–48, 2013.

[39] A. Jabbar Altaay, S. Bin Sahib, and M. Zamani, *An Introduction to Watermarking Techniques Recent Advances in Electrical and Computer Engineering*, WSEAS, Greece, 2013.

[40] A. Jabbar Altaay, S. Bin Sahib, and M. Zamani, *Multimedia Data Hiding Evaluation Metrics Recent Researches in Information Science and Applications*, WSEAS, Greece, 2013.

[41] A. Jabbar Altaay, S. Bin Sahib, and M. Zamani, "Partial smooth region photos statistical steganalysis," *International Journal of Information Processing and Management*, vol. 4, no. 7, pp. 130–143, 2013.

[42] A. Jabbar Altaay, S. Bin Sahib, and M. Zamani, *Pixel Correlation Behavior in Different Themes Communications in Computer and Information Science*, Vol. 398, Springer, Berlin, Heidelberg, Germany, 2013.

[43] A. Jabbar Altaay, S. Bin Sahib, and M. Zamani, "Statistical steganalysis of four photo themes before embedding," in *Proceedings of the International Conference on Advanced Computer Science Applications and Technologies*, Sarawak, Malaysia, December 2013.

[44] A. Jabbar Altaay, S. Bin Sahib, and M. Zamani, "Wide flat region photos statistical steganalysis," *Journal of Convergence Information Technology*, vol. 8, no. 16, pp. 1–14, 2013.

[45] A. Jabbar Altaay, S. Bin Sahib, and M. Zamani, "Wide smooth region photos statistical steganalysis," *Journal of Next Generation Information Technology*, vol. 4, no. 9, pp. 43–56, 2013.

[46] R. Aghbabaeyan, S. Abdullah, and M. Zamani, "Review of digital watermarking techniques," in *Proceedings of the 1st Technology Education and Science International Conference*, Skudai, Malaysia, July 2013.

[47] M. Salem Atoum, S. Ibrahim, G. Sulong, and M. Zamani, "A new method for audio steganography using message integrity," *Journal of Convergence Information Technology*, vol. 8, no. 14, pp. 35–44, 2013.

[48] Z. S. Shams Dolatabadi, A. Abdul Manaf, and M. Zamani, "Using three levels DWT to increase robustness against geometrical attacks," *International Journal of Advancements in Computing Technology*, vol. 5, no. 14, pp. 86–104, 2013.

[49] F. Tohidi, A. Abdul Manaf, M. Zamani, and H. Jamshidi, "Improving the capacity of watermarking techniques by using block truncation coding," *International Journal of Digital Content Technology and Its Applications*, vol. 7, no. 14, pp. 33–47, 2013.

[50] Z. Su, G. Zhang, F. Yue, L. Chang, J. Jiang, and X. Yao, "SNR-constrained heuristics for optimizing the scaling parameter of robust audio watermarking," *IEEE Transactions on Multimedia*, vol. 20, no. 10, pp. 2631–2644, October 2018.

[51] B. Lei, F. Zhou, E.-L. Tan et al., "Optimal and secure audio watermarking scheme based on self-adaptive particle swarm optimization and quaternion wavelet transform," *Signal Processing*, vol. 113, pp. 80–94, 2015.

Hindawi
Journal of Computer Networks and Communications
Volume 2019, Article ID 4754615, 7 pages
https://doi.org/10.1155/2019/4754615

Research Article

Spectral Expansion Method for Cloud Reliability Analysis

K. Kotteswari [1] and A. Bharathi [2]

[1]*Department of Computer Science and Engineering, Annai Mira College of Engg & Techn., Vellore, India*
[2]*Department of Information Technology, Bannari Amman Institute of Technology, Sathyamangalam, India*

Correspondence should be addressed to K. Kotteswari; kottikarthikeyan17@gmail.com

Received 28 May 2019; Accepted 6 August 2019; Published 2 September 2019

Guest Editor: Mazdak Zamani

Cloud computing is a computing hypothesis, where a huge group of systems is linked together in private, public, or hybrid network, to offer dynamically amendable infrastructure for data storage, file storage, and application. With this emerging technology, application hosting, delivery, content storage, and reduced computation cost are achieved, and it acts as an essential module for the backbone of the Internet of Things (IoT). The efficiency of cloud service providers (CSP) could be improved by considering significant factors such as availability, reliability, usability, security, responsiveness, and elasticity. Assessment of these factors leads to efficiency in designing a scheduler for CSP. These metrics also improved the quality of service (QoS) in the cloud. Many existing models and approaches evaluate these metrics. But these existing approaches do not offer efficient outcome. In this paper, a prominent performance model named the "spectral expansion method (SPM)" evaluates cloud reliability. The spectral expansion method (SPM) is a huge technique useful in reliability and performance modelling of the computing system. This approach solves the Markov model of cloud service providers (CSP) to predict the reliability. The SPM is better compared to matrix-geometric methods.

1. Introduction

Cloud computing is a computing model for enabling expedient, on-demand network service to a pool of computing resources such as servers, storage, networks, services, and applications that can be promptly planned and released with minimum executive effort or service provider interface. Cloud computing has novel characteristics such as service-oriented computing, huge scale resource sharing, and on-demand services [1, 2]. Many cloud applications are developed using multitier architecture [3]. In the traditional model, one-tier, two-tier, 3-tier, to N-tier models are available in the cloud to provide n no. of services based on the tier type. Among these models, Multiple-tier or N-tier cloud is a significant model because it can provide N no. of service together in a single-tier model. As it provides multiple services, the maintenance of performance is more required to this model. To maintain the capacity of the model, the reliability evaluation is necessary for this multitier cloud system. For example, Microsoft Azure Cloud service in [4] and Amazon cloud service (EC2) in [5] uses the multitier cloud environment for their web application. Compared to traditional architecture, the multitier cloud model guarantees the service level agreement (SLA) with combined service in a single physical machine. To manage the SLA, the reliability of the system must be predicted. A multitier model can simultaneously manage multiple users with a high workload. To reduce the usage of power and storage, the main concept called virtualization is evolved in each tier for better performance of the system. In [6], the author comes out with the outcome that multitier architecture is the best suitable with virtualization technology. This combination can save power, storage, and reliability.

Although various studies have proposed approaches recently on maintaining availability and balancing power performance [7–10], no one considered reliability, another major factor of cloud service. Reliability is the assertion that cloud provisions are free from software faults, hardware faults, and other faults that break down the system's efficiency. Reliability modelling is necessary for the cloud environment. In [11], the author evaluates the reliability model

for maintaining the performance and power of the cloud virtualization environment. The reliability maintenance of the system indirectly affects performance and energy. In [12], a quality model called CLOUDQUAL is proposed to manage QoS among cloud service. The quality factors considered in this model are availability, reliability, usability, security, and elasticity. From this, reliability is a significant factor to achieve QoS in a cloud environment. In all existing studies, the reliability is calculated in a different way and considered as an important parameter for QoS of the cloud.

In this work, the high-level performance modelling called the spectral expansion method is proposed to estimate the reliability of the cloud computing environment. The spectral expansion method is a mathematical performance technique for analysis of two-dimensional Markov process of finite state space. This Markov model occurs commonly in reliability and performance problem of computing systems. The Markov model of the computing system is first represented in matrix form to compute the eigenvalues λ and eigenvectors Ψ. The eigenvalues and vectors form a linear equation ν_i and solve it. This solution of the linear equation ν_i determines the reliability of the system in graphical form. The remaining section of the paper is described as follows: Section 2 discusses existing studies; Section 3 explains the background of multitier cloud environment; Section 4 describes the spectral expansion model and algorithm; Section 5 illustrates numerical analysis and comparison between other models; and, finally, Section 6 concludes with a conclusion and future work.

2. Related Work

Reliability and availability are a critical requirement for cloud services and must be represented with appropriate planning and modelling [13, 14]. In [15], the reliability model is focused on a virtual machine (VM) of the cloud environment to accomplish high performance. Vishwanath and Nagappan [16] considered the reliability parameter to maintain the consistency of the cloud-based hardware system. The quality of service (QoS) is improved using the reliability metric in performance analysis. In [17], the author states that reliable service and availability services are strongly associated with the performance analysis of cloud-based service.

A reliability optimization algorithm [18] is introduced to maintain the performance of the top banking sector developed using a cloud system. Xia et al. [19] determined the reliability analysis of the computing system using a non-Markov stochastic Petri net (NMSPN). This method takes service invocation and message as model input. Reliability modelling is done for predicting the fault tolerance of computers. A small change in design or insight may cause high variation in performance modelling. To prevent the variation, reliability modelling is estimated in this work [20]. Peng and Huang [21] propose the reliability framework that collaborates failure dependencies as well as considers individual services and failure source. Reliability of the computing system [22] prevents failure and dependency from occurring. Heimann et al. [23] compute the dependability analysis of the computer system which combines concepts like maintainability, availability, and reliability. System reliability measures the instance of offensive events in the computing system. From this instance, the prevention measure can be applied. In [24], a survey has been presented on reliability and energy efficiency in the cloud computing system. They predict that key challenge in the cloud is to manage the reliability of the system. Reliability is defined in the context of resource failure, in the context of VM failure, in the context of service failure, or in the context of security. Nachiappan et al. [25] have used reliability for cloud storage scheduling in big data. From the existing studies, very few reliability modelling is proposed for a cloud-based system. These studies consider only a few parameters and do not produce efficient outcomes.

In this paper, the spectral expansion method (SPM) is evolved for estimation of reliability modelling. This method has been used for performance modelling of the network-based system. The main task of SPM is to solve any class of the Markov model. This method provides a better solution compared to the matrix-geometric method [26]. Chakka [27] used the spectral expansion technique for finding the solution for the Markov model in some finite queue. This model exacts solution for analysis of the Markov process. The results include the performance analysis of the computing system. The spectral expansion method is applied for the performance and dependability analysis of the computing system. In this work, the SPM is applied to predict the reliability of cloud-based service.

3. Background

In this section, virtualization technology and multitier architecture designed are described.

3.1. Virtualization Technology. In recent years, the virtualization technology has been well developed in business and academics. Virtualization technology desires to provide background where VMs can be operated effectively and share the resource to another host efficiently. This leads to improvement in QoS with minimized cost [28]. Each VM handles whole software development including operating system, middleware, and application. Virtualization also appends the abstraction layer to computer architecture for multiplexing resources among other VMs. Several VMs can be virtualized into a physical system based on its capacity. The VMs can be arranged in series or parallel or distributed according to user's needs. The VMs provide many services such as application service, database service, security service, and software service. Based on the architectural design, each VM acts as a particular cloud service provider. By virtualization of VM into a physical system [29], we can provide a lot of service within a system. VM behaviour [30] is monitored using a virtual monitor machine. The VM can also facilitate physical resources to the operation and isolate the environment for execution of virtualization. VM can also reduce cost consumption, power consumption [31], and

workloads [32, 33]. Therefore, VM hosts as a logical machine that is similar to the real computer system.

3.2. Multitier Architecture. Partitioning the system architecture into layers is a fundamental approach to addressing the complexity, and the order of magnitude increases the rate of change. The system is constructed as a continuous sequence of layers, each of which is independent of adjacent ones and communicates with the previous tier via interfaces. The multitier application is a scalable architecture [6]. The architecture design of the multitier application cloud is good compared [34] to 3-tier application although the implementation complexity is more compared to 3-tier. In 3-tier architecture, it is limited and only three fundamental components or tiers such as presentation layer, business layer, and data layer are used to provide services, but in multitier, it is not limited. It includes these three fundamental tiers and can add many components for providing service. In 3-tier, security component is part of the presentation layer, but in multitier, security tier is created as an independent layer to provide secure applications. Like this, N-component and their services can be integrated into multitier application to provide more application and services to the client with less cost and high performance [35]. In 3-tier application, the user interfaces are less interactive compared to the multitier application. It provides more scalability and availability of services compared to the 3-tier application. Due to more interaction, it provides more on-demand services. This on-demand service is well suited in cloud computing design. Thus, the multitier application is well suited for scheduling and provisioning cloud application in the cloud.

The layers deployed in multitier architecture are as follows: The presentation layer provides an interface between user and cloud server and maintains the user interface between the client and the application server. In the multitier pattern, the presentation layer provides more interaction to the end-user. The security layer provides secure processing of the application to the user. It is a framework that provides authentication and data security. It offers data integrity and data confidentiality services. The applications are validated and authorized to the requested client. The business or domain layer implements client application logic. It applies the application logic. This layer provides all services related to application-specific functionality. The domain layer is executed in one or more application servers which communicate with the security, presentation, and data layers. In each application, a separate virtual machine is deployed. The data layer maintains consistency storage and simplifies access to data stored in persistence storage. The details of each application are stored in the data layer. The applications are developed with the help of information required. The data layer is implemented in one or several database servers. The data layer uses many techniques to store and retrieve data efficiently. Several approaches like caching, master-slave replication, SQL, and NoSQL database are used.

4. Spectral Expansion Method (SPM)

The initial process of evaluating the reliability of the system is to design the Markov model for multitier cloud environment. Consider the multitier cloud architecture as shown in Figure 1, with M identical virtual machines (VMs) and host machines (HMs) processing different and unbounded queue of job. The VM and HM can fail from time to time. The failure may be single and independent or multiple and dependent. In this system design, both single, independent repair and multiple, dependent repair are allowed. In execution of tasks in queue, the VM and HM process a single job at a time and each job requires only one VM processor at a time. The policy applied for the failure of the processor is to resume, repair, and re-sample. The service rate, failure rate, and repair rate follow an exponential distribution.

The system is modelled by SPM. $X(t)$ is the functioning state of the system, denoting the number of VM processors running at time t which varies from 0 to M. $Y(t)$ is the number of processes waiting in the queue system and served processes at time t. The irreducible Markov model representing the multitier cloud system is defined as $I = \{[X(t),\ Y(t)];\ t > 0\}$, with state space $\{0, 1, ..., M\} \times \{0, 1, ...\}$. Tasks are assumed to arrive based on the Poisson process with rate λ when the operative state $X(t) = i$. Two kinds of failures are possible. One is an individual processor breakdown independently at rate δ and is repaired independently at rate θ. The second is an "Overall" breakdown of all current VM and HM processors, at rate δ_0, and the "Overall" repair rate of all current non-processing processors is at θ_M. The processing state of the system never changes during the arrival and departure of the tasks unless if there is an independent attack towards the changes. Hence, the alteration in VM and HM state of the system is done only in matrices P and P_y.

The single-move upward transition is generated by a single task arrival. Hence, Q and Q_y, the single-move upward transition matrices rate, are defined as

$$Q = Q_y = \text{diag}\,[\lambda_0, \lambda_1, \ldots, \lambda_M]. \tag{1}$$

This is applicable for all values of y ($y = 0, 1, ...$).

The single-move downward transition occurs by decampment of the single task that is arrived. R and R_y are the single-move downward matrices rate. Let μ be the service rate of VM and HM processor. The decamping rate of tasks at time t depends on $X(t) = x$ and $Y(t) = y$, and that is defined as $R_y(x, x)$. If $x > y$, then every task is allocated to the processor for service provision and all VM and HM processors are not occupied; thus, the decamping rate $R_y(x, x) = y\mu$. Otherwise if $x \le y$, then all the VM and HM processors are occupied with tasks; therefore, the decamping rate $R_y(x, x) = x\mu$. Thus, we conclude that $R_y(x, x)$ does not rely on y if $y \ge x$ and R_y does not depend upon y if $y \ge M$. Therefore, we have

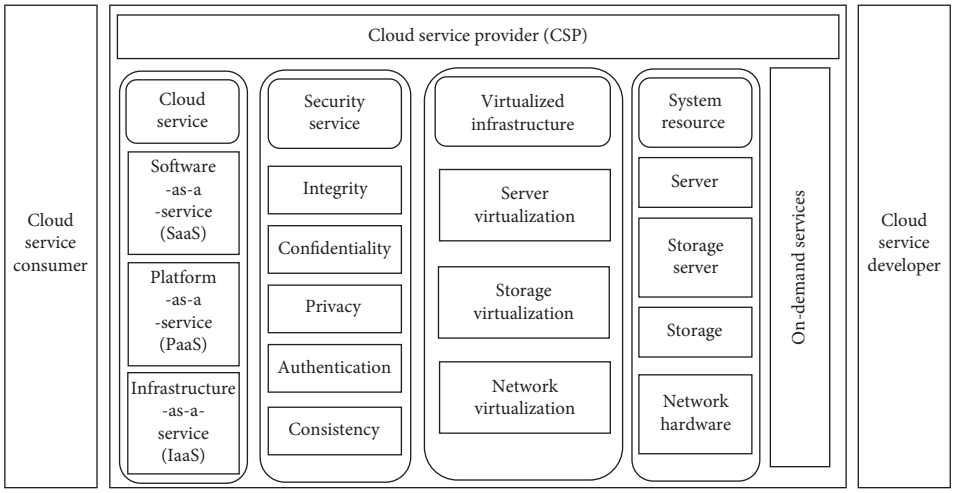


Figure 1: Architecture of the multitier cloud system.

$$R_y = \text{diag}\,[0, \min(y, 1)\mu, \min(y, 2)\mu, \ldots, \min(y, M)\mu], \quad 0 < y < M,$$
$$R = \text{diag}[0, \mu, 2\mu, \ldots, M\mu], \quad y \geq M. \tag{2}$$

R_0 is equal to zero.

To evaluate the reliability of the cloud-based system, three different states are considered based on breakdown and repair rate in the form of matrices P and P_y. The three states are as follows.

Case 1. In this state, VM and HM processors lead to failure independently at a rate of δ per processor. Each nonfunctional processor has a repair rate of θ. There is no synchronization between failure or repair of multiple VM and HM processors.

The matrix P_y and P are defined as

$$P = P_y\ (y = 0, 1, \ldots) = \begin{bmatrix} 0 & M\theta & & & \\ \delta & 0 & (M-1)\theta & & \\ & 2\delta & 0 & \ddots & \\ & & \ddots & \ddots & \theta \\ & & & M\delta & 0 \end{bmatrix}. \tag{3}$$

Case 2. (independent failure and repairs as in state 1, and "overall" failure occurring at rate δ_0). All currently functioning processors fail at this rate except the independent failure:

$$P = P_y\ (y = 0, 1, \ldots) = \begin{bmatrix} 0 & M\theta & & & \\ \delta_0 + \delta & 0 & (M-1)\theta & & \\ \delta_0 & 2\delta & 0 & \ddots & \\ \vdots & & \ddots & \ddots & \theta \\ \delta_0 & & & M\delta & 0 \end{bmatrix}. \tag{4}$$

Case 3. It is similar to state 2, but in addition to this, the currently nonfunctioning processor gets repairs simultaneously. This "Overall" repair occurs at θ_M:

$$P = P_y\ (y = 0, 1, \ldots) = \begin{bmatrix} 0 & M\theta & & & \theta_M \\ \delta_0 + \delta & 0 & (M-1)\theta & & \theta_M \\ \delta_0 & 2\delta & 0 & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \theta \\ \delta_0 & & & M\delta & 0 \end{bmatrix}. \tag{5}$$

The threshold limit is given as H, and it should be $H = M$. The spectral expansion model works until $y = H - 1 = M - 1$. The changes occurred in a functioning state is observed in matrix P; it is likely to find a simple form for the total average service rate of multiprocessor and the steady-state distribution of the number of functioning processors. Let u be the marginal distribution of the number of functioning processors. Then,

$$u = (q_0, q_1, \ldots, q_M) = \sum_{y=0}^{\infty} u_y. \tag{6}$$

This is the probability vector of matrix $P - D^P$ and can be acquired by solving the following equation:

$$u\left(P - D^{P}\right) = 0,$$
$$u\varepsilon = 1. \tag{7}$$

Then, the total average service rate, named as the capacity of multiprocessor service, is $uR\varepsilon$, and average task arrived is $uQ\varepsilon$. It can be emphasized that the cloud-based system is stable when the average task arrival should be less than the capacity of service δ:

$$uQ\varepsilon < uR\varepsilon. \tag{8}$$

5. Numerical Analysis

The reliability of the cloud-based system is implemented using SHARPE tool. The SHARPE tool is an analysis tool in which user-designed architecture is developed and analysis based on spectral Expansion mechanism is done. In this work, the multitier cloud as shown in Figure 2 is developed for the three types of conditions. Using the above mathematical mechanism, the reliability of the multitier cloud for three types of conditions is evaluated. To compare and evaluate the performance of the state, the following parameters are considered such as the capacity of service and the processors with the same service rate. The overall failure rate, δ_0, of state 2 is reimbursed by increasing the value of θ. In type 3, δ_0 is reimbursed by θ_M. The task arrival rate, the capacity of the processing system, and ergodicity condition are the same for all the three states. The three states are compared to show the performance evaluation among them, and the arrival rate is independent for VM and HM processors. Here, the system with 10 processors is considered for the implementation process.

In Figure 3, the effect of the number of tasks executed $E(Y)$ and the arrival rate of the task are described. In Cases 2 and 3, the repair rate and failure rate of VM and HM processors are more compared to Case 1. This fact causes the differentiation of processing time between them. The processing time required for Case 1 is more than Cases 2 and 3. The failure rate of Cases 2 and 3 and the repair rate of Case 3 tend to increase the variance of processing time compared to Case 1. As a result, the reliability of the cloud system in Case 3 is better compared to Case 2 and Case 1. Due to the simultaneous failure and repair rate maintained in Case 3, the average number of jobs execution is increasing compared to other cases.

In Figure 4, the number of processors $M(Y)$ and their service rate are illustrated. For a small range of $N(Y)$, Cases 2 and 3 show improving result better than Case 1, whereas for large $M(Y)$, Case 1 shows better results. The possible explanation for this is that (i) the processor availability reduces when both failure and repair rates increase, (ii) optimal value of M is larger for Case 1 than for Cases 2 and 3. The proposed reliability modelling using SPM is compared with other standard reliability models such as reliability graph model [36] and hierarchical correlation model (HCM) [11]. These two models evaluate reliability in the cloud-based system. In

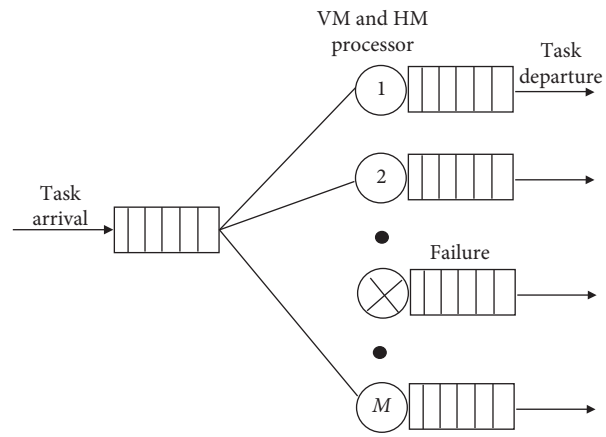


Figure 2: Multiprocessor working mechanism.

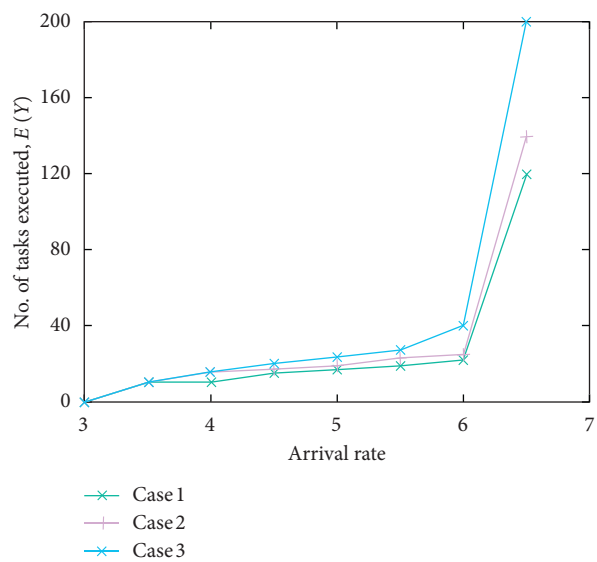


Figure 3: Reliability analysis.

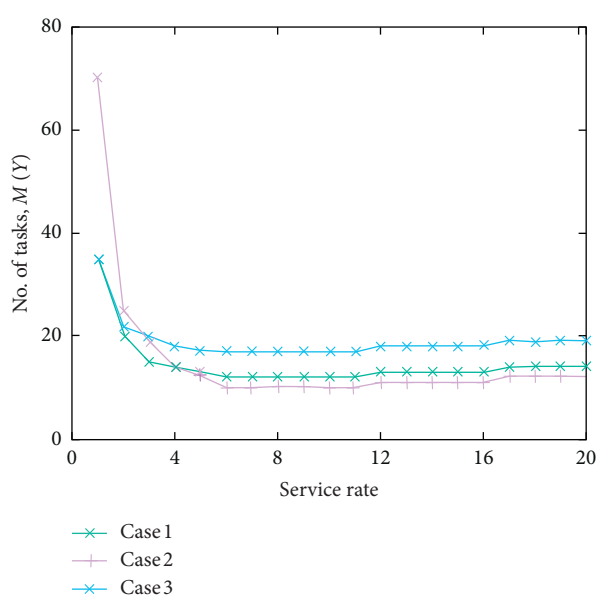


Figure 4: Service rate vs $M(Y)$.

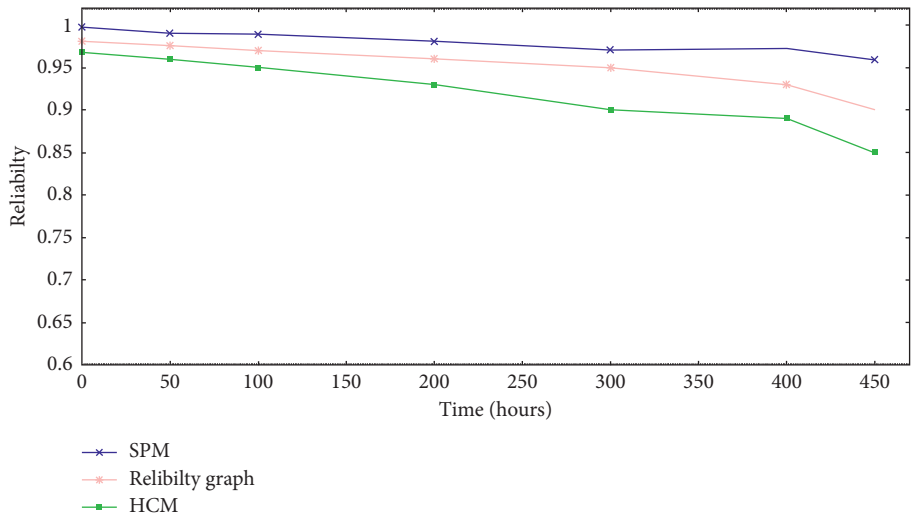


Figure 5: Comparison with other reliability models.

this comparison, the three-reliability model is applied to multitier cloud system and measurements are taken individually. These measurements are drawn as graph comparison in Figure 5. The figure illustrates that the proposed reliability model is efficient than the other two models.

The SPM reliability model maintains accuracy in evaluating reliability for a long duration compared with other models. It gives 0.999997% accurate result compared to other models. Analysing the reliability of multitier cloud is a complex task. The SPM reliability model can analyse this complex task with best accuracy. But other models show less accuracy compared with the proposed model.

6. Conclusion

In this, the reliability of the multitier cloud system is evaluated using the spectral expansion method (SPM). The SPM is developed for three cases of the multitier cloud system. Their reliability modelling is done using SHARPE tool. The comparison of the performance concludes that the system with simultaneous repair and failure rate works with better reliability. Thus, using the SPM, the multitier cloud-based system can be evaluated and analysed. The SPM provides accurate reliability evaluation compared to other standard methods because it considers all parameters for calculating the model. Thus, the graph results show the reliability can be maintained with failure and service rate in processing tasks and also the SPM model can produce accurate result compared to other models. The future work is to implement the SPM reliability model and test on real cloud-based system.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599–616, 2009.

[2] P. Barham, B. Dragovic, K. Fraser et al., "Xen and the art of virtualization," in *ACM SIGOPS Operating Systems Review*, vol. 37, pp. 164–177, no. 5, ACM, Newyork, NY, USA, 2003.

[3] W. Lloyd, S. Pallickara, O. David, J. Lyon, M. Arabi, and K. Rojas, "Performance implications of multi-tier application deployments on infrastructure-as-a-service clouds: towards performance modeling," *Future Generation Computer Systems*, vol. 29, no. 5, pp. 1254–1264, 2013.

[4] S. Turkarslan, *Technical Article for SQL Server and Azure, Application Patterns and Development Strategies for SQL Server in Azure Virtual Machines*, 2014, http://msdn.microsoft.com/en-us/library/azure/dn574746.aspx#comparison.

[5] M. Tavis and P. Fitzsimons, *Web Application Hosting in the AWS Cloud*, Site Point, Melbourne, Australia, 2012.

[6] Y. Diao, J. L. Hellerstein, S. Parekh, H. Shaikh, and M. Surendra, "Controlling quality of service in multi-tier web applications," in *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems (ICDCS'06)*, p. 25, IEEE, Lisboa, Portugal, July 2006.

[7] E. Gelenbe and R. Lent, "Energy–QoS trade-offs in mobile service selection," *Future Internet*, vol. 5, no. 2, pp. 128–139, 2013.

[8] D. Kliazovich, P. Bouvry, and S. U. Khan, "DENS: data center energy-efficient network-aware scheduling," *Cluster Computing*, vol. 16, no. 1, pp. 65–75, 2013.

[9] Y. Xia, M. Zhou, X. Luo, S. Pang, and Q. Zhu, "A stochastic approach to analysis of energy-aware DVS-enabled cloud datacenters," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 1, pp. 73–83, 2015.

[10] G. Nalinipriya, K. G. Maheswari, B. Balusamy, K. Kotteswari, and A. Kumar Sangaiah, "Availability modeling for multi-tier cloud environment," *Intelligent Automation & Soft Computing*, vol. 23, no. 3, pp. 485–492, 2017.

[11] X. Qiu, Y. Dai, Y. Xiang, and L. Xing, "A hierarchical correlation model for evaluating reliability, performance, and

power consumption of a cloud service," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 46, no. 3, pp. 401–412, 2016.

[12] X. Zheng, P. Martin, K. Brohman, and L. D. Xu, "CLOUDQUAL: a quality model for cloud services," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1527–1536, 2014.

[13] N. Kryvinska, C. Strauss, and Z. Peter, "A model to provide higher services availability for the mission critical applications," in *Proceedings of the International Conference on Ambient Systems, Networks and Technologies (ANT-2010), in Conjunction with iiWAS2010*, vol. 8–10, pp. 221–229, Paris, France, November 2010.

[14] N. Kryvinska and C. Strauss, "Conceptual model of business services availability vs. interoperability on collaborative IoT-enabled eBusiness platforms," in *Internet of Things and Intercooperative Computational Technologies for Collective Intelligence*, pp. 167–187, Springer, Berlin, Germany, 2013.

[15] Y. S. Dai, B. Yang, J. Dongarra, and G. Zhang, "Cloud service reliability: modeling and analysis," in *Proceedings of the 15th IEEE Pacific Rim International Symposium on Dependable Computing*, pp. 1–17, Shanghai, China, November 2009.

[16] K. V. Vishwanath and N. Nagappan, "Characterizing cloud computing hardware reliability," in *Proceedings of the 1st ACM Symposium on Cloud Computing*, pp. 193–204, ACM, Indianapolis, IN, USA, June 2010.

[17] E. Bauer and R. Adams, *Reliability and Availability of Cloud Computing*, John Wiley & Sons, Hoboken, NJ, USA, 2012.

[18] Y. Liu, W. Liu, J. Song, and H. He, "An empirical study on implementing highly reliable stream computing systems with private cloud," *Ad Hoc Networks*, vol. 35, pp. 37–50, 2015.

[19] Y. Xia, X. Luo, L. Jia, and Q. Zhu, "A petri-net-based approach to reliability determination of ontology-based service compositions," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 43, no. 5, pp. 1240–1247, 2013.

[20] W. G. Bouricius, W. C. Carter, D. C. Jessep, P. R. Schneider, and A. B. Wadia, "Reliability modeling for fault-tolerant computers," *IEEE Transactions on Computers*, vol. C-20, no. 11, pp. 1306–1311, 1971.

[21] K.-L. Peng and C.-Y. Huang, "Reliability analysis of on-demand service-based software systems considering failure dependencies," *IEEE Transactions on Services Computing*, vol. 10, no. 3, pp. 423–435, 2017.

[22] A. Zhou, S. Wang, B. Cheng et al., "Cloud service reliability enhancement via virtual machine placement optimization," *IEEE Transactions on Services Computing*, vol. 10, no. 6, pp. 902–913, 2017.

[23] D. I. Heimann, N. Mittal, and K. S. Trivedi, "Availability and reliability modeling for computer systems," in *Advances in Computers*, vol. 31, pp. 175–233, Elsevier, Amsterdam, Netherlands, 1990.

[24] Y. Sharma, B. Javadi, W. Si, and D. Sun, "Reliability and energy efficiency in cloud computing systems: survey and taxonomy," *Journal of Network and Computer Applications*, vol. 74, pp. 66–85, 2016.

[25] R. Nachiappan, B. Javadi, R. N. Calheiros, and K. M. Matawie, "Cloud storage reliability for Big Data applications: a state of the art survey," *Journal of Network and Computer Applications*, vol. 97, pp. 35–47, 2017.

[26] I. Mitrani and R. Chakka, "Spectral expansion solution for a class of Markov models: application and comparison with the matrix-geometric method," *Performance Evaluation*, vol. 23, no. 3, pp. 241–260, 1995.

[27] R. Chakka, "Spectral expansion solution for some finite capacity queues," *Annals of Operations Research*, vol. 79, pp. 27–44, 1998.

[28] M. Armbrust, A. Fox, R. Griffith et al., "Above the clouds: a berkeley view of cloud computing," Technical Report No. UCB/EECS-2009-28, Electrical Engineering and Computer Sciences, University of California, Berkeley, CA, USA, 2009.

[29] J. Schad, J. Dittrich, and J.-A. Quiané-Ruiz, "Runtime measurements in the cloud: observing, analyzing, and reducing variance," *Proceedings of the VLDB Endowment*, vol. 3, no. 1-2, pp. 460–471, 2010.

[30] S. Fu, "Failure-aware resource management for high-availability computing clusters with distributed virtual machines," *Journal of Parallel and Distributed Computing*, vol. 70, no. 4, pp. 384–393, 2010.

[31] Y. Ding, X. Qin, L. Liu, and T. Wang, "Energy efficient scheduling of virtual machines in cloud with deadline constraint," *Future Generation Computer Systems*, vol. 50, pp. 62–74, 2015.

[32] W. Zhang, J. Liu, C. Liu, Q. Zheng, and W. Zhang, "Workload modeling for virtual machine-hosted application," *Expert Systems with Applications*, vol. 42, no. 4, pp. 1835–1844, 2015.

[33] X. Xu, H. Jin, S. Wu, and Y. Wang, "Rethink the storage of virtual machine images in clouds," *Future Generation Computer Systems*, vol. 50, pp. 75–86, 2015.

[34] M. S. Rehman and M. F. Sakr, "Initial findings for provisioning variation in cloud computing," in *Procedings of the 2010 IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom)*, pp. 473–479, IEEE, Indianapolis, IN, USA, December 2010.

[35] X. Bu, R. Jia, and C.-Z. Xu, "A reinforcement learning approach to online web systems auto-configuration," in *Proceedings of the 2009 29th IEEE International Conference on Distributed Computing Systems (ICDCS'09)*, pp. 2–11, IEEE, Montreal, Canada, June 2009.

[36] T. A. Nguyen, D. Min, E. Choi, and T. D. Tran, "Reliability and availability evaluation for cloud data center networks using hierarchical models," *IEEE Access*, vol. 7, pp. 9273–9313, 2019.

Hindawi
Journal of Computer Networks and Communications
Volume 2019, Article ID 5747136, 14 pages
https://doi.org/10.1155/2019/5747136

Review Article

A Systematic Literature Review of Authentication in Internet of Things for Heterogeneous Devices

Sanaz Kavianpour,[1] Bharanidharan Shanmugam,[2] Sami Azam,[2] Mazdak Zamani,[3] Ganthan Narayana Samy,[1] and Friso De Boer[2]

[1]*Razak Faculty of Technology and Informatics, Universiti Teknologi Malaysia, Kuala Lumpur 54100, Malaysia*
[2]*College of Engineering, Information Technology and Environment, Charles Darwin University, Darwin, Northern Territory 0909, Australia*
[3]*School of Arts and Sciences, Felician University, Lodi, New Jersey 07644, USA*

Correspondence should be addressed to Bharanidharan Shanmugam; bharanidharan.shanmugam@cdu.edu.au

Received 5 April 2019; Revised 21 July 2019; Accepted 31 July 2019; Published 29 August 2019

Academic Editor: Roberto Nardone

Internet of Things (IoT) has become one of the most significant technologies in recent years because of possessing the diverse application domains. The variety of applications results in a large amount of users' private information diffusion that will pose a paramount security concern. User authentication is a significant factor in the IoT environment as it allows the user to communicate with the device securely. Integration of authentication technologies with IoT ensures secure data retrieval and robust access control. This paper provides a comprehensive systematic literature review of various authentication mechanisms for IoT security proposed in the literature. With the comparison of existing authentication mechanisms that are developed for the IoT in terms of security via a multicriteria classification, the open issues that require further research are identified.

1. Introduction

Internet of Things (IoT) is a new paradigm where everyday objects are interconnected and communicate with each other over the Internet [1]. IoT facilitates a direct integration of physical objects with the cyber world through smart sensors, RFID tags, smartphones, and wearable devices [2]. IoT networks offer various application domains encompassing environmental monitoring, healthcare, smart cities, military affairs, and intelligent transportation system [3, 4]. IoT application will improve rapidly. Cisco systems predict that by 2020, there will be over 50 billion connected things in the Internet consist of sensors, actuators, GPS devices, mobile devices, and all other smart things [5]. The security and privacy of these devices are the most notable challenges in IoT [6]. These devices have an insufficient mechanism of computing platforms, and the communications are wireless often that prone the system to various attacks. Furthermore, the number of devices revealed to the public network are increasing gradually and the devices have direct interaction with the physical world to gather data. All these make them a suitable target for malicious users. Hence, it is substantial to assure devices' authenticity to ensure that the legal device is operating in an expected status and is not affected by malware. As IoT devices are built on various technologies such as power management and sensors, their security requirements vary from one application to another. Several security requirements, which are required to be considered in designing an authentication protocol in IoT perimeter, are delineated in Figure 1.

IoT development is based on wireless networks that collect information for authorised users. In a wireless network, the instructions are sent to terminal nodes by the platform, and the information is collected and transmitted to the platform by the terminal nodes [7]. Mutual authentication is required for the communication process to ensure the security of the network. It hinders illegal adversaries to use the network for malicious tasks. Moreover, other nodes

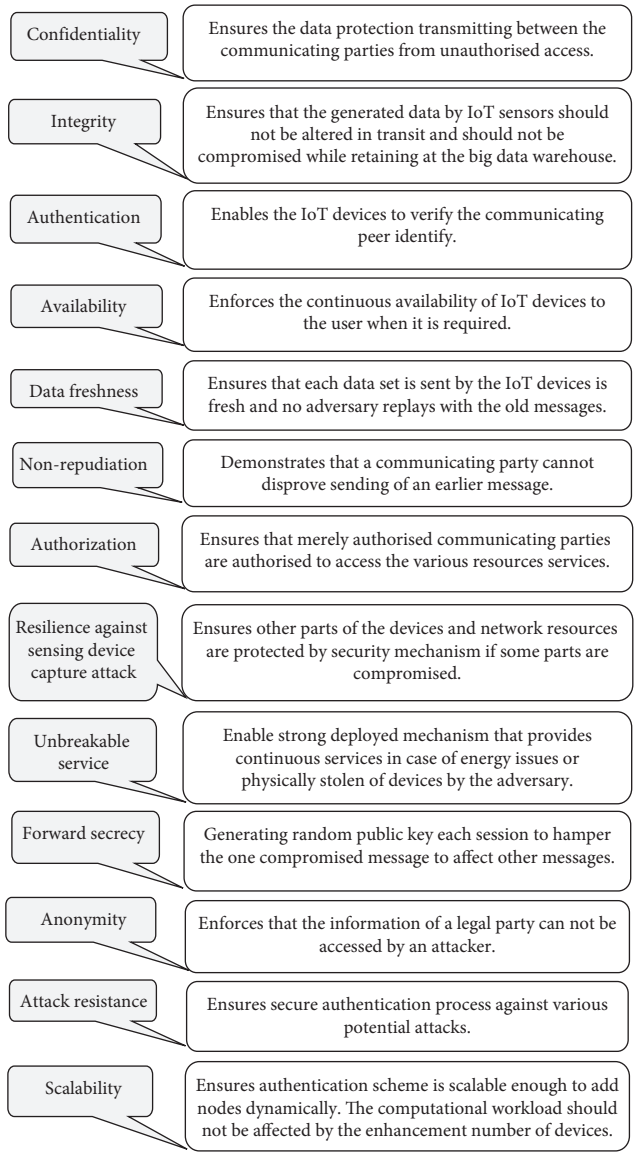


Figure 1: Security requirements in IoT perimeter [6].

should authenticate the terminal nodes to protect the sensor network from being added invalid terminal nodes by the attacker. Mutual authentication has an outstanding role in IoT security. In an unprotected IoT perimeter, the connection of a remote user to other nodes is possible by gaining access to IoT services via smart device applications. Specific information can be extracted from specific nodes once connected. Hence, remote user authentication is vital as inserting resourceful gateway nodes in IoT networks facilitates data delivery and considers most of the processing [8]. In IoT networks, nodes are resource-constrained in terms of processing power, battery backup, memory, speed, and so on.

Authentication factors are ownership, knowledge, and biometrics [9]: ownership factors such as smart cards and smartphones; knowledge factors such as passwords; inherence factors such as fingerprint. Potential authentication scheme can be achieved by integration of a second factor in regards to biometrics [10]. To name multiple biometrics privileges intricate to copy, impossible to be lost or forgotten, difficult to counterfeit, so on. Biometrics is universal, distinctive, persistent, collectable, and unique [11].

Only authenticated and authorised users should be able to utilize the system to hinder security risks. There are various authentication schemes in wireless mobile communication and wireless sensor networks. For instance, in wireless sensor networks, they are based on elliptic curve cryptography [12], self-certified keys cryptosystem [13], and hash functions [14]. Lightweight security solution, key agreement, mutual authentication, and multifactor authentication are significant requirements for a feasible authentication scheme development [15].

This paper is organized as follows. The applied research methodology is presented in Section 2. Section 3 elaborates the systematic literature review results. The comparisons of authentication schemes for the IoT are discussed in Section 4. Finally, Section 5 concludes the paper.

2. Research Method

A systematic literature review (SLR) is employed to review the existing documents about authentication mechanisms in IoT and discuss the results of the conducted review to conduct further research if it is required. Kitchenham and Charters [16] defined SLR as a means of identifying, evaluating, and interpreting all available research relevant to a particular research question, topic area, phenomenon of interest. da Silva et al. [17] stated incorporating current work in a manner that is fair and seen to be fair is known as a systematic review.

2.1. Research Questions. The research questions addressed by this research are as follows:

RQ1. What are the security threats in the IoT perimeter?

RQ2. What are the IoT authentication schemes security issues/challenges?

RQ3. What kind of authentication scheme (techniques) has been developed or applied for IoT-based architectures?

RQ4. What kind of security evaluation is used for IoT authentication?

2.2. Search Process. The search process is as follows. Primary search process consists of searching research keywords (authentication, CPS, IoT, and lightweight) through search engines such as ACM Digital Library, IEEE Xplore, and ScienceDirect. Secondary search process comprises searching of publications manually in the relevant area of research.

2.3. Inclusion and Exclusion Criteria. The papers that describe research on authentication schemes for IoT are included in our review. Papers with respect to their years,

metrics, techniques, evaluation criteria, and results have been examined. The inclusion of papers was based on the similarity of the research with IoT authentication scheme topic. The papers that did not describe security experimental results were excluded from the review.

3. Results of the Systematic Literature Review

In this section, the result of our SLR is formulated. The results focus the set answers to the questions taken as bottom-line of our systematic literature review. Each subsection provides information to answer these questions regarding the objective of the studies. Different tables are shown to represent the results of this review. Authentication schemes in IoT showing the distribution of studies per year and publication source are investigated.

3.1. RQ1: What Are the Security Threats in the IoT Perimeter? The presence of IoT devices in unprotected perimeters escalates the necessity of considering all possible security threats that can compromise the devices. Security threats in the IoT networks are described in Figure 2.

3.2. RQ2: What Are the IoT Authentication Schemes Security Issues/Challenges? Table 1 illustrates the cloud-driven IoT authentication schemes' security issues/challenges with the proposed solutions [18–22].

3.3. RQ3: What Kind of Authentication Scheme (Techniques) Has Been Developed or Applied for IoT-Based Architectures?

3.3.1. Cloud-Based IoT Authentication. Cloud can be a proper platform for storing and processing the IoT devices data. Cloud computing and IoT integration affect our daily life tasks impressively. Cloud-driven IoT privileges are more than a generic IoT architecture. Processing the real-time queries can be performed with less cost and alleviated processing overhead by cloud-driven IoT [18]. In network architecture, a remote object/user should verify itself within the IoT and cloud architecture. Hence, the authentication scheme is required. Table 2 summarizes the relevant schemes based on IoT-cloud architecture authentication.

A cloud-based platform can be employed as a big data warehouse for IoT data. In IoT-based critical applications, only authorised users can access the IoT sensors data or query stored data on cloud servers to realise the hidden patterns of some phenomena. Wazid et al. [37] discussed the authentication schemes for cloud-driven IoT-based big data environment and provided a comparative study of numerous existing authentication schemes that are shown in Table 3.

3.3.2. Lightweight Authentication. A lightweight and secure authentication scheme is required because of the above-mentioned weaknesses in the IoT-cloud architecture schemes. Feng et al. [44] presented a lightweight mechanism

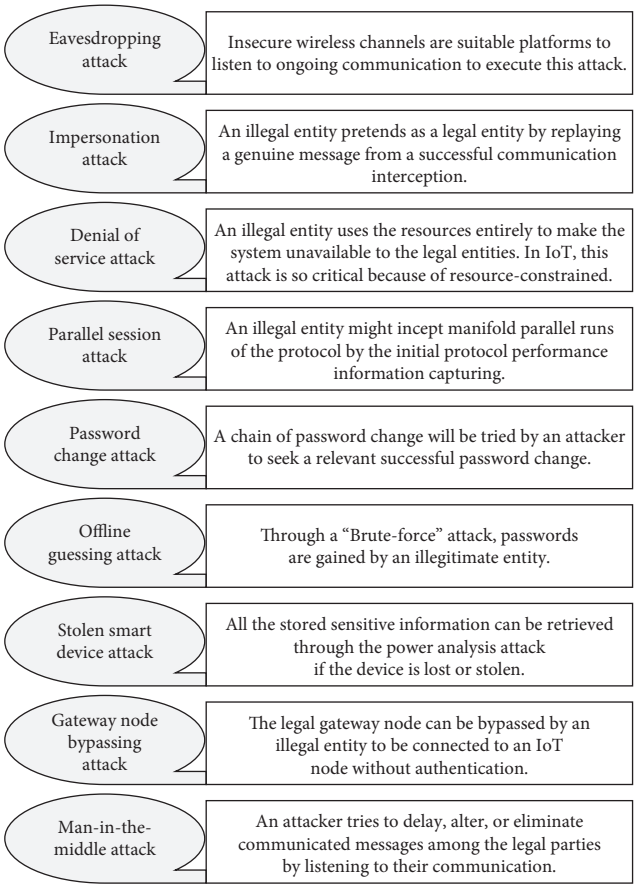


Figure 2: Security threats in the IoT.

for Attestation and Authentication of Things (AAoT), which provides software integrity, mutual authentication, and tamper-proof feature for smart embedded devices. This scheme relies on physical unclonable functions (PUFs). Both strong PUFs and weak PUFs are used by the protocol. PUF-based memory random filling is employed to alleviate the memory resources. The scheme delineates efficient implementations and optimizations for each of the building blocks of AAoT and provides mutual authentication.

(1) *Attestation.* The identity authenticity and software integrity of connected smart embedded devices require to be guaranteed to hinder malicious nodes. The identity authenticity ensures that the device is a legitimate one, and the software integrity ensures the device status and expected behaviors. The identity and integrity of devices can be verified by a known protocol as attestation [45]. Two types of attestation for low-resource devices are software-based attestation [46–49] and attestation based on minimal embedded security architecture [50–52]. These methods are ineffective if IoT devices are impersonated. They are based on the traditional secure storage technology that is costly, rigid, and unsafe [53, 54]. Software-based attestation methods are as follows:

(a) *Time-Based Attestation.* It is the most regular method such as SWATT [46], Pioneer [55], and

Table 1: Security issues/challenges of authentication schemes for cloud-driven IoT and solutions.

Issues/challenges	Description	Solution
Limited computation power and memory storage	The processing capability of IoT devices such as sensors does not have strength in terms of speed. Also, restricted memory storage makes IoT devices incapable of performing operations with high computational power.	Design a user authentication scheme for IoT by employing lightweight cryptographic operations such as the Advanced Encryption Standard (AES) algorithm and cryptographic one-way hash function.
Energy requirement	Some devices have confined battery backup that switches to the power saving mode to save energy when there is no activity. The battery backup restrictions cause difficulty in providing adequate security.	Design a lightweight cryptography authentication scheme.
Scalability	The number of IoT devices enhances gradually.	Design an authentication scheme with the functionality of smart sensing device augmentation.
Mobility	Some devices are mobile such as a wearable sensing device that monitors a person temperature. These devices are connected to different networks based on the user location, which require various security configuration and settings.	Develop a mobility-compliant security technique and interoperability among networks.
Support for heterogeneous devices	Various devices such as a sensor or RFID tags have a different capability of computation, memory, and embedded software.	Design a lightweight authentication scheme.
Dynamic security updates	If a new smart device is added or removed, other entities should be notified through the trusted authority to update this in their memory to prevent security vulnerability.	Employ P2P networks and develop mutual authentication mechanisms.
Protection against physical capturing	If an adversary may physically steal a smart device, the IoT sensor information can be exploited through power analysis attack to perform a malicious task such as replacing this device with another malicious device.	Design the authentication scheme in which in case of a stolen device, the security of other parts does not get affected. Employing tamper-resistant packaging can be useful.
Security and privacy	The stored IoT sensors data at big data warehouse can be used for various kinds of analysis. This data disclosure can compromise data security and privacy.	Apply machine-learning approaches to avoid data leakage.

SCUBA [47]. It performs the checksum computation over the program memory that specifies time delays in case of memory alteration.

(b) *Attestation Based on Memory Filling*. Attackers may elicit the extra memory space. Filling the empty memory by noncompressible pseudo-random noises was proposed by researchers [56–58]. Memory printing [59] and quine [60] can also be used to fill the RAM space. Time-based attestation and attestation based on memory filling are vulnerable to offline static analysis, reverse engineering, and manipulation due to the fixed function.

(c) *Attestation Based on Random Construction of Attestation Function*. Shaneck et al. [61] proposed a random attestation function by a verifier that is sent to a prover per-protocol run. Park and Shin [62] proposed a novel randomized hash function tailored to low-cost CPUs, which was infeasible due to the network bandwidth consumption and complexity. Software-based attestation is not robust against the following attacks: memory copy attack, proxy attack Pioneer [55], an attack via the address translation mechanisms [63], the code compression attack [64], and attack exploiting high execution-time variance [65]. Hence, the software-based attestation security is polemic.

(d) *Embedded Security Architecture and Attestation*. Recent research is based on a hardware-software codesign such as SMART [60] or TrustLite [51]. The goal is to make a dynamic trust anchor in a constrained embedded device. The trust anchor established can be further used to design a scalable collective attestation fulfill security requirements. Previous attestation methods were focused on software attacks.

(2) *Authentication*. Traditional authentication method employs a cryptographic logic with a secret key that lacks secure hardware features and has a high cost. PUFs implemented a secure lightweight device authentication by exploiting the unavoidable manufacturing variations of an integrated circuit to generate a unique device fingerprint. PUF-based

Table 2: IoT-cloud architecture authentication schemes.

Scheme	Method	Weakness
Liao and Wang [23]	Multiserver architecture authentication scheme with the dynamic identity concept.	Vulnerable to user and server forgery attacks.
Hsiang and Shih [24]	The secure dynamic id improvement according to the remote user authentication scheme in a multiserver environment.	Vulnerable to wrong password change, replay attack, and impersonation attack.
Sood et al. [25]	A secure dynamic identity-based authentication protocol for multiserver architecture.	
Lee et al. [26]	A new dynamic identity authentication scheme employed in a multiserver situation.	Vulnerable to forgery attack and stolen-verifier attack.
Li et al. [27]	Employing smart cards in a multiserver architecture.	Vulnerable to impersonation attack, offline password guessing attack, etc.
Leu and Hsieh [28]	Employing smart cards as a secure dynamic id-based remote authentication scheme for distributed systems.	Vulnerable to offline password guessing attack if smart card loss.
Xue et al. [29]	A pseudonym identity-based authentication with key agreement protocol.	Inadequate identity-hidden feature and vulnerable to the offline password guessing attack.
Shunmuganathan et al. [30]	Employing smart-card-based remote user authentication.	Vulnerable to replay, offline password guessing, and impersonation attack.
Zhu [31]	Flexible and password-authenticated key agreement scheme based on chaotic maps for multiple server architecture.	The user tracking attack could affect for multiserver framework.
Li et al. [32]	Using an improved smart card authentication scheme.	Vulnerable to offline password guessing attack.
Irshad et al. [33]	Applying the anonymous-based authentication key agreement in the multiserver architecture.	Vulnerable to user tracking and insider attack.
Maitra et al. [34]	Employing password and smart card: cryptanalysis and design.	Vulnerable to the offline guessing and user tracking attack.
Amin et al. [35]	A lightweight authentication protocol for IoT-enabled devices in a distributed computing environment.	Vulnerable to the offline guessing attack and does not satisfy audit property.
Zhou et al. [36]	Employing a lightweight two-factor authentication scheme that consists of one-way hash function and exclusive-or operation with cloud assistance.	

Table 3: Authentication schemes applicable for cloud-driven IoT-based big data environment.

Scheme	Description	Weakness
Yeh et al. [38]	Employing elliptic curve cryptography (ECC) for user authentication in wireless sensor networks (WSNs).	Not support mutual authentication.
Turkanović et al. [39]	User authentication and key agreement scheme for heterogeneous wireless sensor networks.	Vulnerable to offline password and identity guessing.
Hsieh and Leu [40]	Employing an authentication scheme for WSNs, which can be applicable for IoT as well.	Vulnerable to an insider, offline guessing, user forgery, and sensor node physical capture attack. Not providing session key security and mutual authentication.
Farash et al. [41]	User authentication and key establishment for a heterogeneous architecture of WSNs.	Vulnerable to password guessing in offline mode through the lost/stolen smart card, user impersonation, and session-specific temporary information leakage attack.
Challa et al. [42]	Authentication scheme based on ECC digital signature.	Inadequate computation and communication overheads in IoT.
Li et al. [43]	Authentication scheme based on ECC for industrial IoT.	Requires high computational overhead for the resource-constrained sensor nodes and cannot provide essential functionality features like the biometric update.

authentication strengthens authentication protocols due to the singularity, reproducibility properties, and unclonability. There are two types of PUFs as weak and strong PUFs [66]. Kong et al. [67] presented a PUFatt as a novel PUF design (called ALU PUF) in regards to the delay difference in two different

Table 4: Developed attestation schemes and the drawbacks.

Scheme	Description	Weakness
Kong et al. [67]	PUFatt links software-based attestation to fundamental device characteristics through a novel processor-based PUF, which provide a lightweight remote attestation scheme.	(i) Requires any hardware modifications. (ii) High computation and storage overhead. (iii) No real performance and implementation details of the integrated system. (iv) Does not support mutual authentication. (v) Vulnerable to modeling attacks by machine-learning techniques, and security protocols are unsafe.
Schulz et al. [68]	A lightweight remote attestation scheme that employs PUFs to connect software attestation to remotely identifiable hardware.	(i) Generating additional CRPs enhance the computational and communication difficulty that is not appropriate for low-end devices.

arithmetic and logic units by employing the approach proposed by Schulz et al. [68]. Table 4 describes the relevant developed attestation schemes as well as their drawbacks.

3.3.3. Decentralized Blockchain-Based Authentication. Making an efficient centralized authentication system for IoT is not feasible because of IoT size and other features. Hammi et al. [69] presented a decentralized authentication mechanism called bubbles of trust. They create bubbles in which things can identify and trust each other by using blockchain Ethereum that implements smart contracts. The Master (a device of the bubble) sends a transaction including the Master's identifier and the group identifier. The uniqueness of both will be checked by the blockchain. The bubble will be created if the transaction is valid. In turn, the Followers (each object that makes part of the system) send transactions to the relevant bubbles. Every Follower has a ticket that consists of a groupID, an objectID, pubAddr, and a Signature structure (using elliptic curve digital signature algorithm (ECDSA).The uniqueness of the Follower's identifier and the Follower's ticket validity are verified by the smart contract. In case of an initial successful transaction, there is no need for latter authentication by the Follower.

Various research studies were done on the blockchains and IoT integration while few pieces of researches were on the blockchain's application in meeting IoT security requirements. Table 5 outlines the researches that are mostly based on the security mechanism such as Bitcoin or Ethereum. Although these mechanisms ensure anonymity completely, there is no identification assurance as a fundamental requirement. Private blockchains should be used to ensure identification. These mechanisms have limitations such as the difficulty of inserting a new service or a device. No implementations or simulations were provided by these researches.

3.3.4. Biometrics-Based Remote User Authentication Schemes (Multifactor Authentication). Dhillon and Kalra [79] proposed a lightweight biometric multifactor remote user authentication and key agreement scheme for IoT security. The protocol used a gateway node for the user registration first. Henceforth, the user connects to the sought sensor node by his smart device. The proposed multifactor biometric user authentication phases are as follows:

(1) User Registration Phase. It consists of two phases after IoT deployment as the registration between the user and gateway node and the registration between the gateway node and the sensor node. Registration with the gateway node is required for access to IoT. The user executes the authentication phase to generate a shared session key. The second registration is required to add nodes dynamically to the network.

(2) Login Phase. To initiate the authentication phase, the user should log in to the IoT services. Registration and authentication are based on the user biometrics and password.

(3) Authentication Phase. The user and the node generate an encrypted secret session key that can be used onetime. There is mutual authentication between a remote user and the IoT node.

(4) Password-Change Phase. The user needs to update his/her password periodically for security reasons.

Table 6 indicates various biometrics-based remote user authentication schemes that have been done by researchers. They mostly employ the cryptographic key establishment between the user and the gateway node.

3.4. RQ4: What Kind of Security Evaluation is Used for IoT Authentication? Security of an authentication scheme requires to be evaluated via various security analysis metrics to ensure that is not vulnerable to attacks. Formal security analysis should be performed to verify the scheme security.

3.4.1. Proverif. It is a formal verification tool that checks the security properties' compatibility for cryptographic protocols [96]. Proverif patronages cryptographically operations comprise symmetric encryption/decryption, hash functions, and bit-commitment. Generally, it is employed by researchers to evaluate security reachability, proving session key secrecy, and authentication.

In 2019, Zhou et al. [36] used Proverif to test their scheme security and robustness against common attacks. They employed the two most prevalent cryptotechniques for

Table 5: Security mechanism based on Bitcoin or Ethereum.

Scheme	Description	Weakness
Christidis and Devetsikiotis [70]	Integration of blockchains and smart contracts simplifies the sharing of IoT services and resources as well as authorises the automation cryptographically.	No discussion of the IoT security improvement.
Malviya [71]	Explanation of how blockchain features used for IoT security.	Blockchains type is not specified, and there is no implementation.
Bahga and Madisetti [72]	Proposing an architecture for a blockchain platform for the industrial IoT, which advances the cloud-based manufacturing functionality.	Due to the key-pair generated by the device, system security is low.
Hardjono and Smith [73]	A privacy-preserving method called as ChainAnchor used for cloud and IoT device integration.	Cannot support identification.
Huh et al. [74]	Presenting an approach, which integrates blockchains to IoT and configures each object by a proprietary smart contract.	Both malicious and the nonmalicious user can utilize the system due to the full anonymity of the used objects.
Ruta et al. [75]	Proposing a framework for semantic web of things by a semantic-based resource discovery layer and a basic blockchain infrastructure combination.	Using a private blockchain limits its usage.
Dorri et al. [76]	Developing an architecture that consists of a local blockchain per use case, a shared blockchain, and an overlay blockchain.	The local blockchains limit availability as they are centralized. Moreover, there is an identification issue and communication enhancement because of the high activity of nodes.
Ouaddah et al. [77]	Proposing a framework for IoT (FairAccess) that stores the policies in a private blockchain.	Handling just policy-based systems makes it infeasible for IoT.
Xu et al. [78]	Presenting saphire, which is a blockchain-based distributed storage system to be applied for vast data analytics.	There is no IoT security requirements consideration.

Table 6: Biometrics-based remote user authentication schemes.

Scheme	Description	Weakness
Khan and Zhang [80]	Enhancing the security of a biometric authentication scheme.	Does not provide mutual authentication, and it is vulnerable to server spoofing attacks.
de Meulenaer et al. [81]	Providing comparison on the cost of two key agreement protocols as Kerberos and the Elliptic Curve Diffie–Hellman key exchange with authentication delivered by the elliptic curve digital signature algorithm (ECDH-ECDSA).	It can be applied only where a trusted third party is available. Furthermore, the energy cost of listening increases communication costs.
Li and Hwang [82]	Proposing an authentication scheme based on a biometric and smart card that delivers less computational cost.	Lack of security enhancement.
He et al. [83]	Presenting user authentication and key agreement schemes for WSN.	There is no mutual authentication, and it is not robust against attacks.
Yao et al. [84]	Proposing an electrocardiogram-signal-based protocol using biometric encryption to provide mutual authentication.	An insufficient balance between safety and security.
Huang et al. [85]	Providing an authentication framework considering password, smart card, and biometrics.	Lack of threat identification in the provided framework.
An [86]	Improvement of an effective biometric-based authentication scheme employing smart cards and its analysis in terms of security.	Susceptible to impersonation and gateway node bypassing attack.
Kothmayr et al. [87]	Proposing two-way IoT security architecture in which authentication is provided by datagram transport Layer security handshake.	Supporting just it, DTCT, SSR, and PP partly.
Liu et al. [88]	Presenting an authentication and access control in the IoT.	The high cost of message exchange and inadequate security.
Li et al. [89]	Proposing a multifactor scheme employing biometric, password, and random nonce generated by the user and server.	Insufficient security advancement.

Table 6: Continued.

Scheme	Description	Weakness
Liao and Hsiao [90]	Developing an RFID scheme by ID-verifier transfer protocol combination.	Does not provide tag authentication, privacy, server, and mutual authentication. Moreover, it is vulnerable to tag masquerade, server spoofing, location tracking, and tag cloning attacks.
Xue et al. [91]	Proposing user authentication and key agreement schemes for WSN.	Susceptible to denial-of-service, man-in-the-middle, password guessing, and parallel session attack.
Ndibanje et al. [92]	The protocol provides user anonymity, mutual authentication, and secure session key establishment.	
Saied et al. [93]	Presenting a lightweight collaborative key establishment scheme for the IoT.	
Turkanović et al. [39]	Proposing a user authentication and key agreement scheme for heterogeneous ad hoc WSN.	Vulnerable to offline password guessing and offline identity guessing.
Chen et al. [94]	Developing a two-factor user authentication scheme for WSN with high security.	Vulnerable to replay, impersonation, denial-of-service, and man-in-the-middle attack.
Das and Goswami [95]	Proposing an anonymous biometric-based authentication scheme employing smart cards.	Does not support user anonymity, and it is susceptible to impersonation, password change, parallel session, smart card or device stolen, and gateway node bypassing attack.

Table 7: Performance evaluation.

Scheme	User auditability	Session key for all entities	Mutual authentication	Security
Li et al. [43]	✓	✓	✓	✓
Amin et al. [35]	×	✓	✓	×
Maitra et al. [34]	×	×	✓	×

Table 8: Comparison of authentication schemes based on association phase.

Scheme	Description	Number of messages
Wazid et al. [37]	Proposing an authentication protocol for WSNs using ECC.	5 (if the use of a gateway be required, the number of messages can be duplicated)
Kothmayr et al. [87]	Presenting an authentication scheme employing Data-gram Transport Layer Security (DTLS) algorithm.	5 (can be augmented similar to change cipher suite message and become 8)
Jan et al. [97]	Developing a robust IoT authentication scheme.	4
Hartke and Tschofenig [98]	Presenting an authentication scheme applying DTLS algorithm.	5 (can be appended similar to change cipher suite message and become 8)
Hammi et al. [69]	Proposing a decentralized blockchain-based authentication system for IoT.	2

secure communications as the AES and SHA-2 (256 bits) algorithms. Arduino Uno platform was used with the implementation of SHA-3 (512-bits). The results delineate that their scheme is practical to be implemented for IoT devices. Table 7 illustrates the comparison of this scheme with another two relevant ones. In terms of security and efficiency, [43] delineates superior results than the other two schemes.

Hammi et al. [69] applied Ethereum as blockchain and developed the smart contract to ensure the scheme functionality. They encode and decode Ethereum data for the interactions between end nodes and the blockchain suing a C++ interface. The comparison of [69] with other authentication schemes based on association phase is described in Table 8. The evaluation was based on the execution time, energy consumption, and financial cost. In regards to constrained devices, the fewer number of messages denotes minor system's consumption. If the implementation is on the same hardware, energy, and computation consuming will be alleviated. Additionally, Messages' authentication is realised by the ECDSA. Message authentication required time is based on the employed blockchain compared to other schemes, which is in some milliseconds. This scheme provides a robust identification, authentication, data integrity, and availability. The evaluation results delineate that this scheme ensures the security requirements as well as resiliency toward attacks. This scheme cannot be employed in actual applications, and it has costs due to the cryptocurrency used by the blockchain system.

Table 9: Robustness evaluation against attacks.

Scheme	Mutual Authentication	Key agreement	User anonymity	Password change	Replay attack	Impersonation attack	Password change attack	Parallel session attack	Gateway node bypassing attack	Smart card/smart device stolen attack	Man-in-the-middle attack	Denial-of-service attack
He et al. [83]	×	×	✓	✓	✓	×	×	×	×	×	×	×
An [86]	×	✓	×	×	✓	×	×	✓	×	✓	×	✓
Xue et al. [91]	✓	✓	✓	✓	✓	✓	×	×	✓	✓	×	×
Turkanović et al. [39]	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Chen et al. [94]	✓	×	✓	×	×	×	×	×	×	×	×	×
Das and Goswami [95]	✓	✓	×	✓	✓	×	×	×	×	×	✓	✓
Dhillon and Kalra [79]	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 10: Comparison of evaluation performance.

Scheme	Supported PUF type	CRPs number	Hardware modifications	Implementation	Evaluation on the system	Mutual authentication
Schulz et al. [68]	Strong PUFs	Iteration	✓	×	×	×
Kong et al. [67]	Strong PUFs	Iteration	✓	ALU PUF	×	×
Feng et al. [44]	Strong PUFs and weak PUFs	2	×	SRAM PUF	✓	✓

3.4.2. Burrows–Abadi–Needham (BAN) Logic. A set of rules that can be used to analyze the authentication protocols are known as BAN logic [41]. Through BAN logic, the communicating parties can specify the trustworthiness of the exchanged messages and mutually authenticate each other in IoT perimeter.

3.4.3. Real-or-Random (ROR Model). The ROR model can be used to authenticate key exchange protocols.

3.4.4. Automated Validation of Internet Security Protocols and Applications (AVISPA). A software tool that attests the resilience of an authentication scheme against replay and man-in-the-middle attacks [99]. It analyzes large-scale Internet security protocols and applications. High-Level Protocol Specification Language (HLPSL) is employed to code the protocols.

The proposed authentication scheme by Dhillon and Kalra delineates robustness against numerous attacks that are depicted in Table 9. This scheme provides mutual authentication and a secure key as well as ensures password protection. The scheme security analysis is performed through formal verification using the AVISPA tool, which ensures its security if it is compromised.

3.4.5. Cryptographic Protocol Shapes Analyzer (CPSA). CPSA is an analysis tool comprising authentication tests that count indispensable various cryptographic protocol executions [100]. It is based on a Dolev-Yao model considering an intruder with unlimited access that can specify authentication as well as secrecy properties.

Feng et al. [44] analyzed the AAoT protocol security using the CPSA tool. The experimental results delineate the mutual authentication as well as the secrecy of CRPs and PUFRoT within the authorization protocol. In AAoT, the possibility of memory copy attack is substantial as a valid checksum cannot be counterfeited [44]. AAoT is robust against cloning or impersonating because of the PUF unclonability and unpredictability. A cloned or impersonated prover requires access to the PUF CRPs or keys from the legitimate prover device to fraud verifier.

Table 10 demonstrates the comparison result of AAoT scheme with the most two similar approaches by [67, 68]. The concentration of AAoT is on static attestation. Additional techniques are required for runtime protection such as control flow attestation. AAoT is applicable for low-resource devices. To secure the IoT and CPS, the integration of attestation and authentication can be practical. AAoT ensures either integrity or authenticity and removes the gap from the protocol theory to tangible realization. The AAoT covers the problems in PUFatt.

4. Discussion

The IoT-based perimeters security issues and challenges that are required for authentication techniques are discussed. Furthermore, various existing authentication mechanisms for IoT are reviewed and compared. For instance, in [79] a lightweight authentication scheme uses one-way hash, perceptual hash functions, and XOR operations that are inexpensive. Hence, this scheme is appropriate for resource-constrained IoT devices. Although this scheme depicts robustness against various security attacks through AVISPA tool, it requires to be set up on a testbed to detect the memory requirements and its applicability for real IoT devices.

To secure IoT and CPS, integration of attestation and authentication is so significant and effective. Most of the existing schemes only render one of them. In [44], a lightweight attestation and authentication of low-resource things in IoT and CPS called AAoT are proposed. This scheme presents a feasible and secure hardware-software codesign employing the present resources limited in the smart embedded devices. AAoT integrity and authenticity are achieved by memory-based attestation and PUF-based authentication. In [43], a lightweight IoT-based authentication scheme in cloud computing circumstance presents robustness against different types of attacks and provides mutual authentication. The performance evaluation of this scheme delineates a highly suitable authentication scheme for real IoT-cloud circumstances. A decentralized blockchain-based authentication system for IoT called bubbles of trust is presented in [69]. Devices can communicate securely in the created virtual zones. Despite satisfying security requirements due to using a public blockchain and resilience toward attacks, it has three significant issues as follows: inadaptable to real-time applications, requiring an initial phase, and cryptocurrency rate transformation.

Our findings demonstrate that various types of solutions have been proposed to address the secure authentication scheme for IoT. Even though the existing authentication schemes are applicable in IoT perimeter, there is still a gap, which necessitates additional effort in designing and developing a more secure authentication mechanism to hinder

security breaches. The collected information from this SRL will assist future researchers by providing different directions to cover the gap and design a secure authentication scheme for IoT.

5. Conclusions

A massive heterogeneous network of IoT devices generates a huge amount of data that must be reachable and can be retrieved by only authorised user. Attack surfaces increase swiftly by the connection of billions of IoT devices. To provide secure access to the devices, services, and communication exchanges, authentication is a challenging task. Researchers proposed various authentication schemes, which might be different from each other and applicable across different domains. This paper reviewed different ways to perform authentication in IoT perimeters to identify the challenges and opportunities. The representative security requirements, security issues, and challenges of authentication schemes for IoT have been discussed. Various authentication schemes such as cloud-based IoT, lightweight, decentralized blockchain-based, and biometrics-based remote user authentication (multi-factor) were analyzed. Furthermore, the formal verification analysis provided by the different verification tools comprising Proverif, BAN logic, ROR model, AVISPA, and CPSA for evaluating the security of the authentication schemes is described. Although these authentication schemes for IoT have resulted in more secure design, they still have limitations, which require massive improvement to ensure deeper privacy and security. It is significant that the authentication scheme will be able to provide high security for IoT. Hence, a more in-depth investigation is necessitated in this direction, as achieving a robust authentication scheme is still an open issue.

Conflicts of Interest

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

References

[1] G. Zhao, J. Wang, J. Luo, X. Long, and X. Si, "Applicability of elliptic curve cryptography on internet of things," *Energy Procedia*, vol. 11, pp. 128–133, 2011.

[2] Z. Dawy, W. Saad, A. Ghosh, J. G. Andrews, and E. Yaacoub, "Towards massive machine type cellular communications," *IEEE Wireless Communications Magazine*, vol. 24, no. 1, pp. 120–128, 2017.

[3] X. Zeng, S. K. Garg, P. Strazdins, P. P. Jayaraman, D. Georgakopoulos, and R. Ranjan, "IOTSim: a simulator for analysing IoT applications," *Journal of Systems Architecture*, vol. 72, pp. 93–107, 2017.

[4] C. M. Sosa-Reyna, E. Tello-Leal, and D. Lara-Alabazares, "Methodology for the model-driven development of service oriented IoT applications," *Journal of Systems Architecture*, vol. 90, pp. 15–22, 2018.

[5] D. Evans, "The internet of things, how the next evolution of the internet is changing everything," in *Whitepaper, Cisco Internet Business Solutions Group (IBSG)*, vol. 1, pp. 1–12, 2011, http://www.cisco.com/c/dam/enus/about/ac79/docs/innov/IoTIBSG0411FINAL.pdf.

[6] K. Zhao and L. Ge, "A survey on the internet of things security," in *Proceedings of the 2013 Ninth International Conference on Computational Intelligence and Security*, pp. 663–667, Leshan, China, December 2013.

[7] G. Zhao, X. Si, J. Wang, X. Long, and T. Hu, "A novel mutual authentication scheme for internet of things," in *Proceedings of 2011 International Conference on Modelling, Identification and Control*, Shanghai, China, June 2011.

[8] M. Henze, L. Hermerschmidt, D. Kerpen, R. Häußling, B. Rumpe, and K. Wehrle, "A comprehensive approach to privacy in the cloud-based internet of things," *Future Generation Computer Systems*, vol. 56, pp. 701–718, 2016.

[9] Y. Choi, J. Nam, D. Lee, J. Kim, J. Jung, and D. Won, "Security enhanced anonymous multiserver authenticated key agreement scheme using smart cards and biometrics," *The Scientific World Journal*, vol. 2014, Article ID 281305, 15 pages, 2014.

[10] C.-H. Lin and Y.-Y. Lai, "A Flexible biometrics remote user authentication scheme," *Computer Standards & Interfaces*, vol. 27, no. 1, pp. 19–23, 2004.

[11] W.-C. Kuo, H.-J. Wei, Y.-H. Chen, and J.-C. Cheng, "An enhanced secure anonymous authentication scheme based on smart cards and biometrics for multi-server environments," in *Proceedings of the 10th Asia Joint Conference on Information Security*, Kaohsiung, Taiwan, May 2015.

[12] Z. Benenson, N. Gedicke, and O. Raivio, "Realizing robust user authentication in sensor networks," in *Real-World Wireless Sensor Networks (REALWSN)*, 2005.

[13] C. Jiang, B. Li, and H. Xu, "An efficient scheme for user authentication in wireless sensor networks," in *Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07)*, pp. 438–442, Niagara Falls, Canada, May 2007.

[14] X. H. Le, M. Khalid, R. Sankar, and S. Lee, "An efficient mutual authentication and access control scheme for wireless sensor networks in healthcare," *Journal of Networks*, vol. 6, no. 3, pp. 355–364, 2011.

[15] T.-H. Chen and W.-K. Shih, "A robust mutual authentication protocol for wireless sensor networks," *ETRI Journal*, vol. 32, no. 5, pp. 704–712, 2010.

[16] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Keele University, Keele, UK, EBSE 2007-001, 2007.

[17] F. Q. B. da Silva, M. Suassuna, R. F. Lopes et al., "Replication of empirical studies in software engineering: preliminary findings from a systematic mapping study," in *Proceedings of the 2011 Second International Workshop on Replication in Empirical Software Engineering Research*, Banff, Canada, September 2011.

[18] G. S. Aujla, R. Chaudhary, N. Kumar, A. K. Das, and J. J. P. C. Rodrigues, "SecSVA: secure storage, verification, and auditing of big data in the cloud environment," *IEEE Communications Magazine*, vol. 56, no. 1, pp. 78–85, 2018.

[19] A. K. Das, S. Zeadally, and D. He, "Taxonomy and analysis of security protocols for internet of things," *Future Generation Computer Systems*, vol. 89, pp. 110–125, 2018.

[20] Y. Yang, H. Peng, L. Li, and X. Niu, "General theory of security and a study case in internet of things," *IEEE Internet of Things Journal*, vol. 4, no. 2, pp. 592–600, 2017.

[21] J. Singh, T. Pasquier, J. Bacon, H. Ko, and D. Eyers, "Twenty security considerations for cloud-supported internet of

things," *IEEE Internet of Things Journal*, vol. 3, no. 3, pp. 269–284, 2016.

[22] S. M. R. Islam, D. Kwak, M. H. Kabir, M. Hossain, and K.-S. Kwak, "The internet of things for health care: a comprehensive survey," *IEEE Access*, vol. 3, pp. 678–708, 2015.

[23] Y.-P. Liao and S.-S. Wang, "A secure dynamic id based remote user authentication scheme for multi-server environment," *Computer Standards & Interfaces*, vol. 31, no. 1, pp. 24–29, 2009.

[24] H.-C. Hsiang and W.-K. Shih, "Improvement of the secure dynamic id based remote user authentication scheme for multi-server environment," *Computer Standards & Interfaces*, vol. 31, no. 6, pp. 1118–1123, 2009.

[25] S. K. Sood, A. K. Sarje, and K. Singh, "A secure dynamic identity based authentication protocol for multi-server architecture," *Journal of Network and Computer Applications*, vol. 34, no. 2, pp. 609–618, 2011.

[26] C.-C. Lee, T.-H. Lin, and R.-X. Chang, "A secure dynamic id based remote user authentication scheme for multi-server environment using smart cards," *Expert Systems with Applications*, vol. 38, no. 11, pp. 13863–13870, 2011.

[27] X. Li, Y. Xiong, J. Ma, and W. Wang, "An efficient and security dynamic identity based authentication protocol for multi-server architecture using smart cards," *Journal of Network and Computer Applications*, vol. 35, no. 2, pp. 763–769, 2012.

[28] J.-S. Leu and W.-B. Hsieh, "Efficient and secure dynamic id-based remote user authentication scheme for distributed systems using smart cards," *IET Information Security*, vol. 8, no. 2, pp. 104–113, 2014.

[29] K. Xue, P. Hong, and C. Ma, "A lightweight dynamic pseudonym identity based authentication and key agreement protocol without verification tables for multi-server architecture," *Journal of Computer and System Sciences*, vol. 80, no. 1, pp. 195–206, 2014.

[30] S. Shunmuganathan, R. D. Saravanan, and Y. Palanichamy, "Secure and efficient smart-card-based remote user authentication scheme for multiserver environment," *Canadian Journal of Electrical and Computer Engineering*, vol. 38, no. 1, pp. 20–30, 2015.

[31] H. Zhu, "Flexible and password-authenticated key agreement scheme based on chaotic maps for multiple servers to server architecture," *Wireless Personal Communications*, vol. 82, no. 3, pp. 1697–1718, 2015.

[32] X. Li, J. Niu, S. Kumari, J. Liao, and W. Liang, "An enhancement of a smart card authentication scheme for multi-server architecture," *Wireless Personal Communications*, vol. 80, no. 1, pp. 175–192, 2015.

[33] A. Irshad, H. F. Ahmad, B. A. Alzahrani, M. Sher, and S. A. Chaudhry, "An efficient and anonymous chaotic map based authenticated key agreement for multi-server architecture," *KSII Transactions on Internet and Information Systems*, vol. 10, no. 12, pp. 5572–5595, 2016.

[34] T. Maitra, S. K. H. Islam, R. Amin, D. Giri, M. K. Khan, and N. Kumar, "An enhanced multiserver authentication protocol using password and smart-card: cryptanalysis and design," *Security and Communication Networks*, vol. 9, no. 17, pp. 4615–4638, 2016.

[35] R. Amin, N. Kumar, G. P. Biswas, R. Iqbal, and V. Chang, "A light weight authentication protocol for IoT-enabled devices in distributed cloud computing environment," *Future Generation Computer Systems*, vol. 78, pp. 1005–1019, 2018.

[36] L. Zhou, X. Li, K.-H. Yeh, C. Su, and W. Chiu, "Lightweight IoT-based authentication scheme in cloud computing circumstance," *Future Generation Computer Systems*, vol. 91, pp. 244–251, 2019.

[37] M. Wazid, A. K. Das, V. Odelu, N. Kumar, and W. Susilo, "Secure remote user authenticated key establishment protocol for smart home environment," *IEEE Transactions on Dependable and Secure Computing*, 2017.

[38] H.-L. Yeh, T.-H. Chen, P.-C. Liu, T.-H. Kim, and H.-W. Wei, "A secured authentication protocol for wireless sensor networks using elliptic curves cryptography," *Sensors*, vol. 11, no. 5, pp. 4767–4779, 2011.

[39] M. Turkanović, B. Brumen, and M. Hölbl, "A novel user authentication and key agreement scheme for heterogeneous ad hoc wireless sensor networks, based on the internet of things notion," *Ad Hoc Networks*, vol. 20, pp. 96–112, 2014.

[40] W.-B. Hsieh and J.-S. Leu, "A robust user authentication scheme using dynamic identity in wireless sensor networks," *Wireless Personal Communications*, vol. 77, no. 2, pp. 979–989, 2014.

[41] M. S. Farash, M. Turkanović, S. Kumari, and M. Hölbl, "An efficient user authentication and key agreement for heterogeneous wireless sensor network tailored for the internet of things environment," *Ad Hoc Networks*, vol. 36, pp. 152–176, 2016.

[42] S. Challa, M. Wazid, A. K. Das et al., "Secure signature-based authenticated key establishment scheme for future IoT applications," *IEEE Access*, vol. 5, pp. 3028–3043, 2017.

[43] X. Li, J. Niu, M. Z. A. Bhuiyan, F. Wu, M. Karuppiah, and S. Kumari, "A robust ECC-based provable secure authentication protocol with privacy preserving for industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 8, pp. 3599–3609, 2018.

[44] W. Feng, Y. Qin, S. Zhao, and D. Feng, "AAoT: lightweight attestation and authentication of low-resource things in IoT and CPS," *Computer Networks*, vol. 134, pp. 167–182, 2018.

[45] T. Abera, N. Asokan, L. Davi et al., "Invited—things, trouble, trust: on building trust in IoT systems," in *Proceedings of the 53rd Annual Design Automation Conference (DAC'16)*, pp. 1–6, Austin, TX, USA, June 2016.

[46] A. Seshadri, A. Perrig, L. van Doorn, and P. Khosla, "SWATT: software-based attestation for embedded devices," in *Proceedings of the IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004*, pp. 272–282, Berkeley, CA, USA, May 2004.

[47] A. Seshadri, M. Luk, A. Perrig, L. van Doorn, and P. K. Khosla, "SCUBA: secure code update by attestation in sensor networks," in *Proceedings of the 5th ACM workshop on Wireless Security*, pp. 85–94, New York, NY, USA, September 2006.

[48] F. Armknecht, A. R. Sadeghi, S. Schulz, and C. Wachsmann, "A security framework for the analysis and design of software attestation," in *Proceeding of the 2013 ACM SIGSAC Conference on Computer & communications Security*, pp. 1–12, Berlin, Germany, November 2013.

[49] J. Horsch, S. Wessel, F. Stumpf, and C. Eckert, "SobTrA: a software-based trust anchor for ARM cortex application processors," in *Proceedings of the 4th ACM Conference on Data and Application Security and Privacy*, pp. 273–280, San Antonio, TX, USA, March 2014.

[50] K. El Defrawy, A. Francillon, D. Perito, and G. Tsudik, "SMART: secure and minimal architecture for (establishing a dynamic) root of trust," in *Proceedings of the 19th Annual*

Network and Distributed System Security Symposium, San Diego, CA, USA, Febuary 2012.

[51] P. Koeberl, S. Schulz, A.-R. Sadeghi, and V. Varadharajan, "TrustLite: a security architecture for tiny embedded devices," in *Proceedings of the Ninth European Conference on Computer Systems (EuroSys' 14)*, pp. 1–14, Amsterdam, Netherlands, April 2014.

[52] F. Brasser, B. El Mahjoub, A.-R. Sadeghi, C. Wachsmann, and P. Koeberl, "TyTAN: tiny trust anchor for tiny devices," in *Proceeding 52nd ACM/EDAC/IEEE Design Automation Conference (DAC '15)*, pp. 1–6, San Francisco, CA, USA, June 2015.

[53] R. Maes, "PUF-based entity identification and authentication," in *Physically Unclonable Functions*, pp. 117–141, Springer, Berlin, Heidelberg, 2013.

[54] S. Zhao, Q. Zhang, G. Hu, Y. Qin, and D. Feng, "Providing root of trust for ARM trust zone using on-chip SRAM," in *Proceedings of the 4th International Workshop on Trustworthy Embedded Devices (TrustED '14)*, pp. 25–36, Scottsdale, Arizona, USA, November 2014.

[55] A. Seshadri, M. Luk, E. Shi, A. Perrig, L. van Doorn, and P. Khosla, "Pioneer: verifying code integrity and enforcing untampered code execution on legacy systems," in *Proceedings of the Twentieth ACM Symposium on Operating Systems Principles (SOSP '05)*, pp. 1–16, Brighton, UK, October 2005.

[56] T. AbuHmed, N. Nyamaa, and D. Nyang, "Software-based remote code attestation in wireless sensor network," in *Proceedings of the GLOBECOM 2009—2009 IEEE Global Telecommunications Conference*, pp. 1–8, Honolulu, HI, USA, November 2009.

[57] Y. Yang, X. Wang, S. Zhu, and G. Cao, "Distributed software-based attestation for node compromise detection in sensor networks," in *Proceedings of the 26th IEEE International Symposium on Reliable Distributed Systems (SRDS) 2007*, pp. 219–230, Beijing, China, October 2007.

[58] D. Perito and G. Tsudik, "Secure code update for embedded devices via proofs of secure erasure," in *Proceedings of the European Symposium on Research in Computer Security (ESORICS) 2010*, pp. 643–662, Athens, Greece, September 2010.

[59] M. Jakobsson and K.-A. Johansson, "Practical and secure software-based attestation," in *2011 Workshop on Security and Privacy, 2011 Lightweight Security & Privacy: Devices, Protocols, and Applications*, pp. 1–9, Istanbul, Turkey, March 2011.

[60] V. Gratzer and D. Naccache, "Alien vs. quine, the vanishing circuit and other tales from the industry's crypt," in *Proceedings of the 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'06)*, pp. 48–58, St. Petersburg, Russia, May 2006.

[61] M. Shaneck, K. Mahadevan, V. Kher, and Y. Kim, "Remote software-based attestation for wireless sensors," in *Proceedings of the Second European Conference on Security and Privacy in Ad-Hoc and Sensor Networks, 2005 (ESAS'05)*, pp. 27–41, Visegrad, Hungary, July 2005.

[62] T. Park and K. G. Shin, "Soft tamper-proofing via program integrity verification in wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 4, no. 3, pp. 297–309, 2005.

[63] G. Wurster, P. C. van Oorschot, and A. Somayaji, "A generic attack on check summing-based Software tamper resistance, security and privacy," in *Proceedings of the 2005 IEEE Symposium on Security and Privacy (S&P'05)*, pp. 127–138, Oakland, CA, USA, May 2005.

[64] C. Castelluccia, A. Francillon, D. Perito, and C. Soriente, "On the difficulty of software-based attestation of embedded devices," in *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS '09)*, pp. 400–409, Chicago, IL, USA, November 2009.

[65] Y. Li, Y. Cheng, V. Gligor, and A. Perrig, "Establishing software-only root of trust on embedded systems: facts and fiction," in *Security Protocols XXIII*, pp. 50–68, Springer, Berlin, Germany, 2015.

[66] U. Ruhrmair and D. E. Holcomb, "Pufs at a glance," in *Proceedings of the Conference on Design, Automation & Test in Europe (DATE '14)*, Dresden, Germany, March 2014.

[67] J. Kong, F. Koushanfar, P. K. Pendyala, A.-R. Sadeghi, and C. Wachsmann, "PUFatt: embedded platform attestation based on novel processor-based PUFs," in *Proceedings of the 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, San Francisco, CA, USA, June 2014.

[68] S. Schulz, A.-R. Sadeghi, and C. Wachsmann, "Short paper: lightweight remote attestation using physical functions," in *Proceedings of the Fourth ACM Conference on Wireless Network Security (WISEC '11)*, pp. 109–114, Hamburg, Germany, June 2011.

[69] M. T. Hammi, B. Hammi, P. Bellot, and A. Serhrouchni, "Bubbles of trust: a decentralized blockchain-based authentication system for IoT," *Computers & Security*, vol. 78, pp. 126–142, 2018.

[70] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.

[71] H. Malviya, "How blockchain will defend IoT," http://www.slideshare.net/HiteshMalviya/how-blockchain-will-defend-iot, 2016.

[72] A. Bahga and V. K. Madisetti, "Blockchain platform for Industrial Internet of things," *Journal of Software Engineering and Applications*, vol. 9, no. 10, 2016.

[73] T. Hardjono and N. Smith, "Cloud-based commissioning of constrained devices using permissioned blockchains," in *Proceedings of the 2nd ACM International Workshop on IoT Privacy, Trust, and Security (IoTPTS '16)*, pp. 29–36, Xi'an, China, May 2016.

[74] S. Huh, S. Cho, and S. Kim, "Managing IoT devices using blockchain platform," in *Proceedings of the 19th International Conference on Advanced Communication Technology (ICACT)*, Bongpyeong, Republic of Korea, Febuary 2017.

[75] M. Ruta, F. Scioscia, S. Ieva, G. Capurso, A. Pinto, and E. Di Sciascio, "A blockchain infrastructure for the semantic web of things," in *Proceedings of the SEBD 2018: 26th Italian Symposium on Advanced Database Systems*, Castellaneta Marina, Italy, June 2018.

[76] A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram, "Blockchain for IoT security and privacy: the case study of a smart home," in *Proceedings of the 2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, Kona, HI, USA, March 2017.

[77] A. Ouaddah, A. A. Elkalam, and A. A. Ouahman, "Towards a novel privacy-preserving access control model based on blockchain technology in IoT," in *Proceedings of the Europe and MENA Cooperation Advances in Information and Communication Technologies*, pp. 523–533, Niagara Falls, Canada, October 2017.

[78] Q. Xu, K. M. M. Aung, Y. Zhu, and K. L. Yong, "A blockchain-based storage system for data analytics in the internet of things," in *New Advances in the Internet of Things*, Springer, Berlin, Germany, 2018.

[79] P. K. Dhillon and S. Kalra, "A lightweight biometrics-based remote user authentication scheme for IoT services," *Journal of Information Security and Applications*, vol. 34, pp. 255–270, 2017.

[80] M. K. Khan and J. Zhang, "Improving the security of 'a flexible biometrics remote user authentication scheme'," *Computer Standards & Interfaces*, vol. 29, no. 1, pp. 82–85, 2007.

[81] G. de Meulenaer, F. Gosset, F.-X. Standaert, and O. Pereira, "On the energy cost of communication and cryptography in wireless sensor networks," in *Proceedings of the 2008 IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*, Avignon, France, October 2008.

[82] C.-T. Li and M.-S. Hwang, "An efficient biometrics-based remote user authentication scheme using smart cards," *Journal of Network and Computer Applications*, vol. 33, no. 1, pp. 1–5, 2010.

[83] D. He, Y. Gao, S. Chan, C. Chen, and J. Bu, "An enhanced two-factor user authentication scheme in wireless sensor networks," *Ad Hoc & Sensor Wireless Networks*, vol. 10, no. 4, pp. 361–371, 2010.

[84] L. Yao, B. Liu, G. Wu, K. Yao, and J. Wang, "A biometric key establishment protocol for body area networks," *International Journal of Distributed Sensor Networks*, vol. 7, no. 1, Article ID 282986, 2011.

[85] X. Huang, Y. Xiang, A. Chonka, J. Zhou, and R. H. Deng, "A generic framework for three-factor authentication: preserving security and privacy in distributed systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 8, pp. 1390–1397, 2011.

[86] Y. An, "Security analysis and enhancements of an effective biometric-based remote user authentication scheme using smart cards," *Journal of Biomedicine and Biotechnology*, vol. 2012, Article ID 519723, 6 pages, 2012.

[87] T. Kothmayr, C. Schmitt, W. Hu, M. Brunig, and G. Carle, "A DTLS based end-to-end security architecture for the internet of things with two-way authentication," in *Proceedings of the 37th Annual IEEE Conference on Local Computer Networks—Workshops*, pp. 956–963, Clearwater, FL, USA, October 2012.

[88] J. Liu, Y. Xiao, and C. L. P. Chen, "Authentication and access control in the internet of things," in *Proceedings of the 32nd International Conference on Distributed Computing Systems Workshops*, Macau, China, June 2012.

[89] C.-T. Li, C.-Y. Weng, and C.-C. Lee, "An advanced temporal credential-based security scheme with mutual authentication and key agreement for wireless sensor networks," *Sensors*, vol. 13, no. 8, pp. 9589–9603, 2013.

[90] Y.-P. Liao and C.-M. Hsiao, "A secure ECC-based RFID authentication scheme integrated with ID-verifier transfer protocol," *Ad Hoc Networks*, vol. 18, pp. 133–146, 2014.

[91] K. Xue, C. Ma, P. Hong, and R. Ding, "A temporal-credential-based mutual authentication and key agreement scheme for wireless sensor networks," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 316–323, 2013.

[92] B. Ndibanje, H.-J. Lee, and S.-G. Lee, "Security analysis and improvements of authentication and access control in the internet of things," *Sensors*, vol. 14, no. 8, pp. 14786–14805, 2014.

[93] Y. B. Saied, A. Olivereau, D. Zeghlache, and M. Laurent, "Lightweight collaborative key establishment scheme for the internet of things," *Computer Networks*, vol. 64, pp. 273–295, 2014.

[94] L. Chen, F. Wei, and C. Ma, "A secure user authentication scheme against smart-card loss attack for wireless sensor networks using symmetric key techniques," *International Journal of Distributed Sensor Networks*, vol. 11, no. 4, pp. 63–73, 2015.

[95] A. K. Das and A. Goswami, "A robust anonymous biometric-based remote user authentication scheme using smartcards," *Journal of King Saud University-Computer and Information Sciences*, vol. 27, no. 2, pp. 193–210, 2015.

[96] M. Abadi, B. Blanchet, and H. Comon-Lundh, "Models and proofs of protocol security: a progress report," in *Proceedings of the International Conference on Computer Aided Verification*, pp. 35–49, Grenoble, France, June 2009.

[97] M. A. Jan, P. Nanda, X. He, Z. Tan, and R. P. Liu, "A robust authentication scheme for observing resources in the internet of things environment," in *Proceedings of the 13th International Conference on Trust, Security and Privacy in Computing and Communications*, Beijing, China, September 2014.

[98] K. Hartke and H. Tschofenig, "A DTLS. 1.2 profile for the internet of things. Draft-ietf-dice-profle-00," https://tools.ietf.org/id/draft-ietf-dice-profile-00.html, 2014.

[99] AVISPA, "The security protocol animator for AVISPA," 2017, http://www.avispa-project.org/.

[100] S. F. Doghmi, J. D. Guttman, and F. J. Thayer, "Searching for shapes in cryptographic protocols," in *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pp. 523–537, Springer, Berlin, Germany, October 2007.

Hindawi
Journal of Computer Networks and Communications
Volume 2019, Article ID 9852472, 8 pages
https://doi.org/10.1155/2019/9852472

Research Article

An Efficient Framework for Sharing a File in a Secure Manner Using Asymmetric Key Distribution Management in Cloud Environment

K. V. Pradeep,[1] **V. Vijayakumar**,[1] **and V. Subramaniyaswamy**[2]

[1]*School of Computing Science & Engineering, VIT, Chennai, India*
[2]*School of Computing, SASTRA University, Thanjavur, India*

Correspondence should be addressed to K. V. Pradeep; pradeep.kv@vit.ac.in

Received 18 February 2019; Revised 29 April 2019; Accepted 19 May 2019; Published 27 June 2019

Guest Editor: Arash H. Lashkari

Cloud computing is a platform to share the data and resources used among various organizations, but the survey shows that there is always a security threat. Security is an important aspect of cloud computing. Hence, the responsibility underlines to the cloud service providers for providing security as the quality of service. However, cloud computing has many challenges in security that have not yet been addressed well. The data accessed or shared through any devices from the cloud environment are not safe because they are likely to have various attacks like Identity Access Management (IAM), hijacking an account or a service either by internal/external intruders. The cryptography places a major role to secure the data within the cloud environment. Therefore, there is a need for standard encryption/decryption mechanism to protect the data stored in the cloud, in which key is the mandatory element. Every cloud provider has its own security mechanisms to protect the key. The client cannot trust the service provider completely in spite of the fact that, at any instant, the provider has full access to both data and key. In this paper, we have proposed a new system which can prevent the exposure of the key as well as a framework for sharing a file that will ensure security (CIA) using asymmetric key and distributing it within the cloud environment using a trusted third party. We have compared RSA with ElGamal and Paillier in our proposed framework and found RSA gives a better result.

1. Introduction

As per NIST definition, cloud computing is the one that enables us to share the configured computing resources on demand over the network, which requires less effort from management or from service provider to provision and release the resources rapidly [1]. It is a new pattern to avail the computational power with higher flexibility and availability at low cost [2]. IaaS, PaaS, and SaaS are the three service models evolved [3–5] and can be deployed as public, private, hybrid, and community.

When IT services are outsourced, there is a need to concern about privacy and security, particularly while moving the data from an organization environment to the cloud environment. The primary motivation is to reduce cost, reducing the responsibility of both security and privacy. All the outsourced services are accountable, and it is the responsibility of an organization to process, monitor, and address the various issues involved in both security and privacy. Along with this, an organization has to focus even on performance, availability, and recovery.

Accessing the data often from the data centers with the high-speed Internet by a software application through devices with mobility in nature is what comprises the term cloud. Using cloud computing, the concept has evolved that any kind of services are provisioned quickly on demand as per user requirements. Services offered by the cloud service providers (CSP) can be categorized into Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), and Infrastructure-as-a-Service (IaaS). Based on the usage of these services from the cloud

comes with its own set of merits and demerits. Few may say that it involves unnecessary risk; others may argue that it is a way to minimize the cost for any kind of start-up business. Accessibility, availability, agility, scalability, and efficiency are the main reasons that make most of the business entities attracted and migrated to cloud.

Note that the concept foretold revolves around data, and whenever we talk about data we must address both security and privacy issues [6–9] involved around it, especially with the rate at which data are enormously out growing in quantity. The lack of a well-defined entrance and egress points or the ability to physically control arises the whole host of security issues like data leakage, backups and recovery, abuse of privileged user access, regulatory requirements, multitenancy issues, and many more. Thus, with many gaps to be filled, more work is required in the area of securing the data [10, 11] to make the concept of cloud more acceptable to cloud service consumers.

Cloud is used for storing the data by many organizations, but the computation is done at cloud service provider (CSP) due to its resource (CPU/memory) elasticity, pay as per use, access to data from any location throughout the world, no requirement of any installation and maintenance, and accessing at most relief from storage burden. Normally, the cloud is deployed as private, public, community, or hybrid. There is not a proper interface that can lead to threats to the cloud.

In cryptography [12–21], key is the vital component in the cryptosystem. It is one of the biggest challenging tasks to manage them because it is the state of being legally responsible for any loss beyond the encryption and decryption process, such as people and flawed policies. Just as in a physical lock, a key is vital for providing access. The question arises when the cloud service providers offer their own encrypted and decryption [22–25] mechanism to protect the data; they provide the key along with data. With the encrypted data and the key lying with the cloud service providers itself, they do act as owners and there is very little that is preventing them for abusing that right.

The rest of the paper is organized and ordered into various sections. In Section 2, all the related works associated with cloud storage are discussed and the methods used to overcome them are also reviewed. In Section 3, the proposed technique for storing the data and sharing the key among multiple users was discussed. In Sections 4 and 5, the experimental results with some simulation and analysis are done. Finally, in the last section, both conclusion and summarization of work are discussed.

2. Related Work

As we all know that the cloud provides the facility to avail software, platform, and infrastructure as services and can be availed as either public, private, or hybrid deployment models, most of the organizations prefer to have both computation and storage as a service from the cloud so that they can be free from storage and computation management, and accessing the data based on demand from anywhere, any time, with low cost and also does not require any installation and maintenance. The only thing that should be taken care is to provide secure interfaces and API's so that no malicious intruders (insider/outsider) can share the infrastructure or can hijack the account which may lead to threats in a cloud [26]. For example, booking a train, flight, or bus ticket or updating PAN card through online is possible through the cloud service transparently [27].

The data which are to be stored in the cloud are encrypted by any chosen cryptography technique so that not even cloud service provider (CSP) can access. Normally, users would like to store the data in the encrypted form, so it is necessary to protect the keys used for encrypting the data. Now, the question arises whether the keys are to be stored with either CSP or customer or any other third party authority. Sometimes the data which are stored in the cloud are to be shared among multiple members of the particular members [28–30] using group key.

In the paper [31], the author provides an interface as a service to manage files from different or multiple cloud service providers and also a tool to monitor, download, or upload throughput and single or multikey user accounts. Data stored in the public cloud environment need more attention if the data are more sensitive. Because customers do not have any control over that environment, even he/she will not know where exactly data will be stored or exist geographically. Therefore, it is needed to use any standard cryptography approach to protect both data and keys used. Managing the keys used to encrypt the user's data becomes an integral part, especially when multiple tenants were using the same cloud environment to share the resources. The importance of key management is not well addressed in the European network and Information Security Agency (ENISA) [32].

There are many standard cryptographic techniques available, and can be chosen one among them to store the data in an encrypted form. There are millions and billions of users using storage as a service, and keys used to encrypt and decrypt need to be protected. Now the problem arises, where the key is to be stored either at customer, provider, or any trusted third party.

In [33, 34], the authors introduce a public key infrastructure certificate scheme to secure the cloud application program to encrypt the key, which also binds the key with files based on File-Id attributes for personal storage. It is easy to maintain and manage the keys to update, but it is required to update the user credentials regularly. Suppose if the user credentials are weak, then the security of file cannot be assured. Here, the author encrypts the stored cloud data using an elliptic curve.

In [35], the author divides the data into several portions and the keys are generated randomly to encrypt each portion of data. In [36, 37], to protect the outsourced data or resources, the author uses overencryption which is a two-layered model; i.e., owners will encrypt the resources or data before uploading it to cloud. Once data are received in the encrypted form, now once again the cloud server will reencrypt it for the second time before other users download it.

Yan and Zhou [38] based on the hypergraphs proposed the key management which is scalable in nature and well

suited for the various groups in an organization for large-scale applications. The key management based on keying hypergraph is a group key management.

3. Proposed Method

The functionalities of key management are to deal with the generation and storage of key along with its distribution among two parties. In cloud security, it plays an important role in the attacks on public key cryptographic algorithms. Here, we will be using a modified Diffie–Hellman distribution scheme for the distribution of keys which are used to encrypt and decrypt the data or files which are stored in a cloud on a secure channel.

Public key cryptographic systems are purely based on one-way functions, which consume less computing power to convert plain text into ciphertext. The reason behind the one-way function is that, in a reasonable amount of time, it is not feasible for anyone to bring back the plain text from the ciphertext (i.e., $C = fxb(P)$ and $P \neq f^{-1}xb(C)$, where- C = ciphertext, P = plain text, and Xb is a public key of party B). Diffie–Hellman key exchange is one of the most used key distribution method usually in a public key distribution system (PKDS) to exchange a piece of information.

3.1. How It Works. Assume two parties (P and Q) are involved in communication to exchange the keys over an insecure channel, and it follows as below:

(1) P and Q chooses a large prime number "n" and a primitive element "a" (both are public) along with the one-way function $Y = f(x) = g^x \bmod n$

(2) Both the parties P and Q select their own random numbers to say "p" and "q" and compute the values "M" and "N," respectively, and send them to each other

(a) P computes $M = g^p \bmod n$ and sends M to Q
(b) Q computes $N = g^q \bmod n$ and sends N to P

(3) Upon receiving, both computes the following:

(i) P computes $S = N^a \bmod n = (g^{pq} \bmod n)$
(ii) Q computes $S = M^b \bmod n = (g^{pq} \bmod n)$

(4) Now both the parties P and Q have the same key which is shared over the insecure communication channel

This communication uses a one-way function, which is extremely hard to get back $x = f^{-1}(y)$, even by knowing the values of M, N, n, and g.

From Figure 1, public key cryptography is used to secure and share a file from a cloud computing environment. The proposed framework consists of 5 entities like cloud storage (CS), cloud key management system (CKMS), certificate authority (CA), data owner, and users.

(1) Data owner is the one who uploads the file into the cloud storage and has full control over it. Each file is identified by its File-Id.

(2) Users are the one, who would like to seek permissions and access the file stored in the cloud.

(3) CKMS plays an important role in cryptography, which provides confidentiality (encryption and decryption), authentication of an entity, data origin authentication, integrity, and digital signatures with the help of CA (certificate authority)

(4) CA is responsible for establishing and vouching for the authenticity of public keys.

(5) CS is simply used to store the files, as per KMS and CA approval.

Initially, the owner of the file, which is to be uploaded in the cloud environment, has to register with CKMS by creating an account with login credentials. The same login credentials can be used even to upload the list of users and their permissions with whom the uploaded file can be shared. Based upon the user login details, the CKMS uses asymmetric encryption to generate both public and private key pair associated with the user details. After receiving the request from the file owner to upload the file in the cloud environment, the CKMS will encrypt it by using the private key of an associated file owner, and then, the encrypted file is stored in the cloud storage. Any certified and a standard asymmetric or symmetric key algorithm can be used, which is not discussed in our paper (e.g., RSA and AES). All the above process is being done by CKMS with the help of the certificate authority (CA) and its working strategy is given in Section 3.2.

3.2. Encryption Process. In the above process, the generated private key (R_k) is used to encrypt the file and then the file is stored in the cloud. The public key (U_k) is divided into two parts by CKMS: one part of it is transmitted to the owner, and rest is kept with CKMS.

Later, at the time of decryption, the whole key is used to decrypt the uploaded file. Algorithm for generating the keys and encrypting the file is given in Figure 2, and its notations with definitions are provided in Table 1.

3.3. How Certificate Authority (CA) Works. Normally, it comprises with registration authority, name server, key generator, and certificate directory. The relationships with these components are illustrated in Figure 3:

(a) Certificate authority: it is responsible to bind the public keys to the distinguished user through signed certificates, managing them with the serial number, revocation of certificate, and also to authenticate the public keys

(b) Name server: unique user names are managed

(c) Registration authority: entities are authorized by unique names

(d) Key generator: it generates public/private key pairs and symmetric keys or passwords

(e) Certificate directory: it is a database used to store/organize/manage all the user certificates and also permits the CA

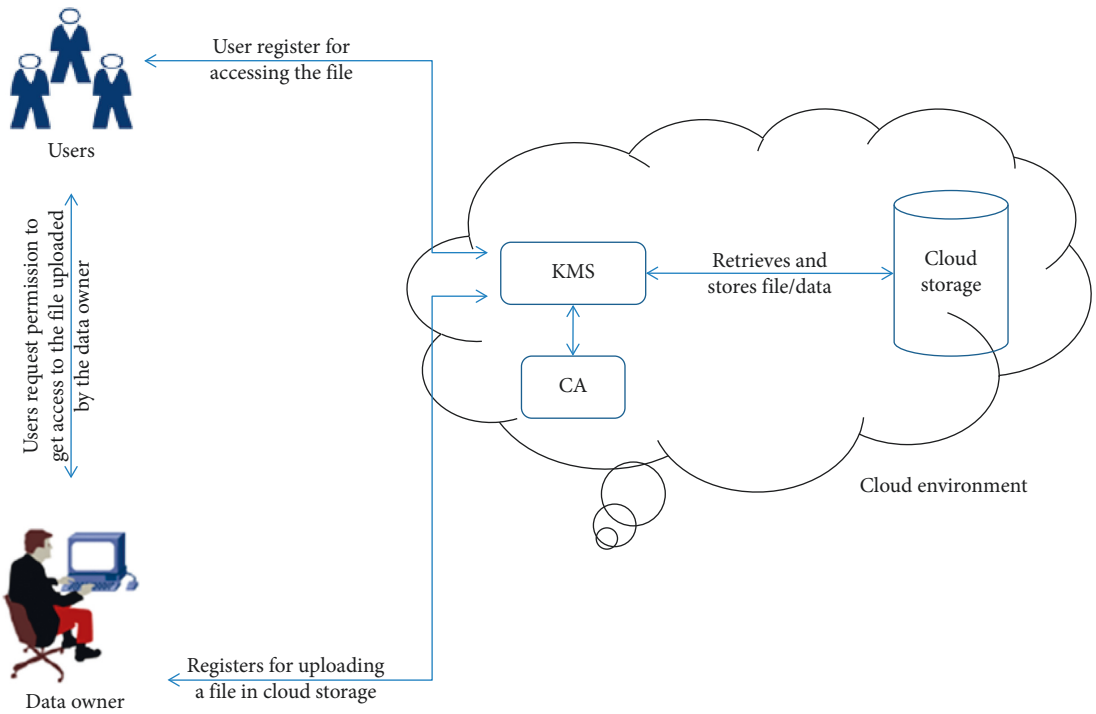


Figure 1: Block diagram of the proposed framework.

Algorithm for encrypting the file by cloud-KMS

Input: file (F), length of the key (K_L), asymmetric encryption (A_{enc})
Output: private key(R_k), public key (U_k), owners key(O_k), KMS key (U_{kms})
If (Owner = TRUE) then do the following
(1) Use A_{enc} to generate U_k and R_k (public and private keys)
(2) $E_F = A_{enc}(F \text{ and } R_k)$ // encryption of file using private key
(3) $U_k = (U_{k1} \text{ and } U_{k2})$ // divides the public keys into 2 parts
(4) $U_0 = U_{k1}$
(5) $U_{kms} = U_{k2}$
(6) DELETE(R_k, U_k) // delete public and private keys
endif.

Figure 2: Encryption process by CKMS.

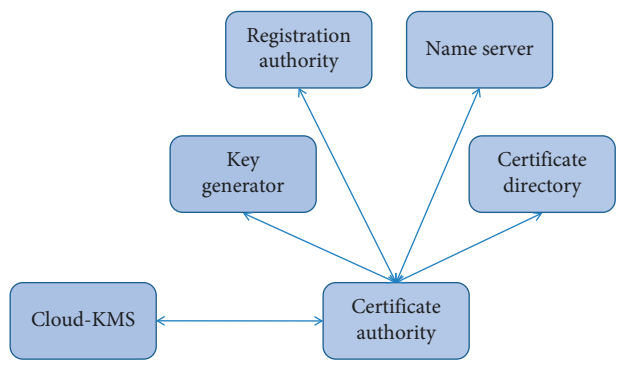


Figure 3: Services related to certificate authority.

Table 1: Notations and definitions.

Notations	Definitions
A_{Enc}	Asymmetric encryption algorithm
U_k	Public key
R_k	Private key
U_{kms}	KMS's part of public key
U_o	Owner's part of public key
E_f	Encrypted file
L_{users}	List of users
APL_{users}	Access permission list
F	File to be uploaded in the cloud
F_{id}	File identifier

3.4. Decryption Process. The file owner shares and uploads the user list and their access permission list with the CKMS and also sends an invitation to all the users who are all in the list to access the file uploaded in the cloud along with the part of owner's public key for self-registration. A user can access the file uploaded, by registering with CKMS. Once registration is completed, the user will receive the login credentials. Upon receiving, the user requests the file through file identifier (F_{id}) along with the part of the owner's public key. Now CKMS checks the user permission details from access permission list associated with file identifier. If the list contains the user information and access permissions, then CKMS regenerates the public key by combining the parts, i.e., the one which received from the user with the part of public key it is having.

During this process, it also downloads the file which is in the encrypted form, from the cloud storage. Now with the generated public key, it decrypts the downloaded file and sends back to the corresponding user. Key translation protocol is used to distribute and exchange the key between users, owner, and CKMS (Figure 4).

Algorithm for decrypting the file by cloud-KMS

Input: F_{id}, E_F, U_{kms}, L_{user}, U_o, APL_{users}, A_{enc}

Output: F_{id}, F

(1) If ((L_{users} = TRUE) && (APL_{users} = TRUE)) then

(2) If (F_{id} = TRUE)

(3) Use A_{enc} to decrypt the File

(4) $U_k \longleftarrow$ concatenate (U_{kms}, U_o)

(5) $F \longleftarrow D\ (F, U_k)$

(6) DELETE (U_k)

(7) Else "File Not Found"

(8) Endif.

Figure 4: Decryption process by CKMS.

Key translation protocol used by cloud-KMS

Summary: A interacts with a KMS (T) and the party B.

Result: A transfers a secret message "M" to B.

(1) Notation: E is a symmetric encryption algorithm

(2) One-time setup: A and T share key K_{AT}. Similarly, B and T Share K_{BT}.

(3) Protocol messages:

(a) $A \longrightarrow T : A, E_{K_{AT}}$ (B, M)

(b) $T \longrightarrow A : E_{K_{BT}}$ (M, A)

(c) $A \longrightarrow B : E_{K_{BT}}$ (M, A)

(4) Protocol actions:

(a) A encrypts M under K_{AT} and sends to T with its own identifier.

(b) Upon decrypting the message, T determines it is intended to B, Look up the key (K_{BT}) of the indicated recipient, and reencrypts M for B.

(c) T returns the translated message for A to send to B.

(d) Finally, A sends it to B.

Figure 5: KT protocol used by CKMS.

CKMS acts as key translation centre (KTC), which allows any two parties (owner to the user) or (CKMS to owner/user), which do not directly share any kind of keying material, to establish a secure communication through the use of long-term keys K_{AT} and K_{BT} with the CKMS. The key translation algorithm is illustrated in Figure 5.

Suppose if the user requesting an uploaded file from cloud to access is not present in the user list, then obviously CKMS will deny the permission.

3.5. Implementation Details. The web-based application is designed to implement the proposed methodology in the cloud environment in OpenStack with the help of JAVA technology. The Graphical User Interface is created and used for self-registration by the owner to access or upload the file, respectively. All users and owners make use of this GUI to register with CKMS and set up the credentials. Upon registrations, file owner will also upload the intended user list and access control or permission list in CKMS using the interface provided. Once the user list is added, these users will get the invitations either from CKMS or owner for self-registration. Now CKMS uses a proposed asymmetric encryption method to encrypt the file which too is uploaded in the cloud environment. Normally, here we have used the RSA-the asymmetric encryption algorithm to generate the keying material (i.e., both public/private keys) and private key is used to encrypt the file which is to be stored in the cloud storage. For all the validations, certification is been verified with certification authority (CA) by CKMS.

The commonly used mechanisms to generate the key in public key infrastructures are Rivest, Shamir, and Adelman (RSA), Diffie–Hellman (DH), digital signature (DS), and so on. The framework which we have proposed performs all the fundamental functionalities of key management such as key distribution and generation, encryption, and decryption, and all are tested on two instances in OpenStack (Virtual Machines) with 4 GB of RAM and a dual-core 1.4 GHz Intel Core i5 processor each. Finally, the performances of the different public key infrastructure algorithms are considered and compared to the process of encryption and decryption, usage of CPU and memory. We carried out experiments on multiple text files with different key sizes like 48 bits, 128 bits, 256 bits, 512 bits, 1024 bits, and 2048 bits.

Since CKMS uses key translation protocol, the parties involved (owner, user, and CKMS) can exchange the messages or keying material more effectively, efficiently in an insecure communication channel.

4. Experimental Results

The time plays a vital role during the generation of key, encryption, and decryption process. All investigations are made on Intel(R) Core-2 with CPU 1.4 GHz processor, 4 GB RAM on Windows-10 working framework by using JAVA. Any algorithm can be measured with the time parameter that the system has consumed. If anyone wants to design an optimized system, it must ensure that the above three in the cryptographic system designed must consume less time. Here, in Figure 6, we have shown the comparative analysis of all the three algorithms of time taken to generate the key. We have found that RSA algorithms consume less time compared to all the other two algorithms irrespective of key size. Here, we have analyzed only time taken to generate the key; likewise, we can even do for time consumed for the encryption and decryption process also because they are the most important to provide the security in a cloud environment which is not focused here in our paper.

The result in Figure 7 shows that the RSA algorithm utilizes fewer resources like CPU, memory, and RAM when compared to ElGamal and Paillier during encryption and decryption of the files stored in the cloud. As we all know, the RSA has limitations while handling the large file (bytes), which consumes more resource utilization. For smaller files, the proposed architecture gives better result by consuming fewer CPU cycles.

Here, we have tested the proposed architecture on the files with different sizes like 48 MB, 64 MB, 128 MB, 256 MB,

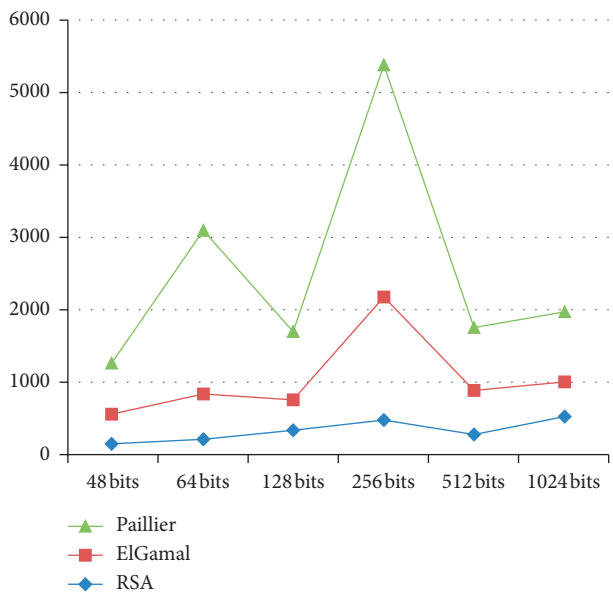


Figure 6: Time consumed to generate a key of different sizes by CKMS used by 3 different algorithms.

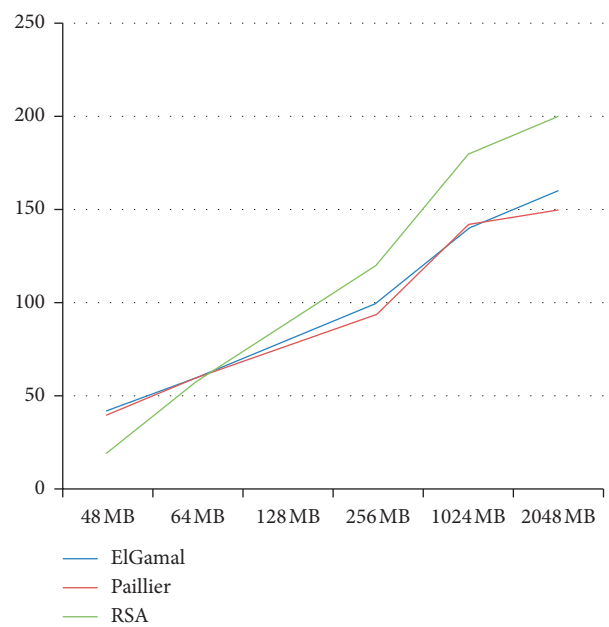


Figure 7: Percentage of CPU usage for different public key algorithms like ElGamal, Paillier, and RSA.

1024 MB, and 2048 MB and the plotted against CPU cycles, as shown in Figure 7.

Apart from CPU usage and time consumption to generate the key, we have investigated the time consumed for encryption and decryption in Figures 8 and 9, respectively. From these figures, we can analyze that RSA consumes less time for encryption and ElGamal consumes less time for the decryption process. ElGamal is usually slower than symmetric ones for the same level of security, so it is faster to encrypt the symmetric key with ElGamal and the message (which can be arbitrarily large) with a symmetric cipher.

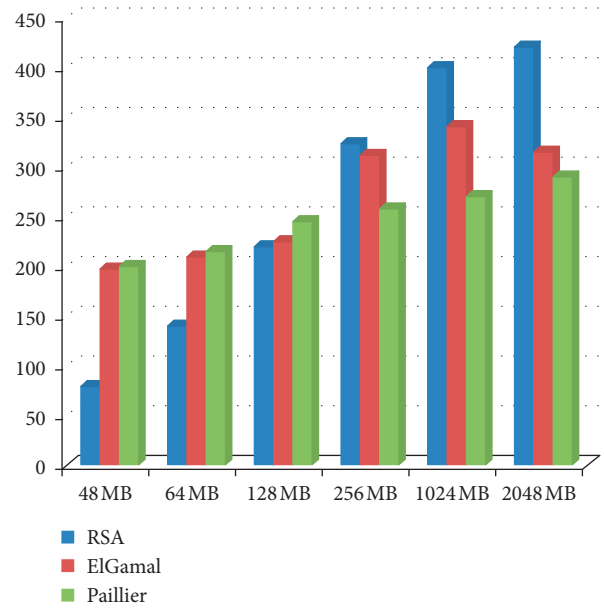


Figure 8: Time consumption during the encryption process.

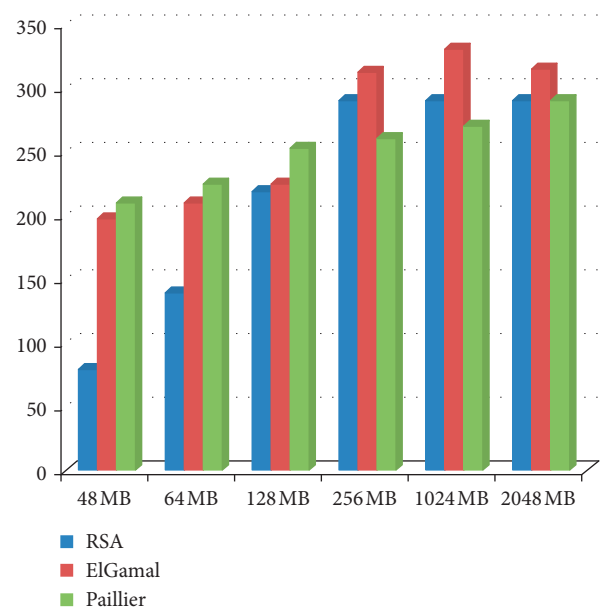


Figure 9: Time consumption during the decryption process.

5. Complexity Analysis

As from the results, we can easily say that RSA is better than ElGamal and Paillier. The complexity analysis w.r.t time for RSA during encryption/decryption is $O(\log N^3)$. If the public "e" is fixed then computing M^e, Mod N has the time complexity $O(\log N^2)$ which is better than other Paillier and ElGamal.

In the encryption, the RSA performs better than ElGamal in all cases. In the decryption process, the ElGamal outperforms RSA, meaning that text messages are decrypted faster by ElGamal than by the RSA technique. When viewed as a single tool, the RSA is superior to the ElGamal algorithm

Table 2: Homomorphic encryption cryptosystems.

Characteristics	RSA	Paillier	ElGamal
Platform	Cloud computing	Cloud computing	Cloud computing
Homomorphic type	Multiplicative	Additive	Multiplicative
Asymmetric in nature	Yes	Yes	Yes
Data privacy	All ensures privacy both in communication and storage (data at rest)		
Data security	Yes	Yes	Yes
Key used by	Separate keys are used for encrypt/decrypt either by customer/provider		
Speed (enc/dec)	Fast/slow	Slow/slow	Slow/fast

in terms of computational speeds. It is the reason why the RSA algorithm has been and is still being used in designing many security protocols for data communication (Table 2).

6. Conclusion and Future Directions

In this paper, we have proposed the framework to share a file in a secure manner using public key infrastructure in the cloud environment to store, share, and download by ensuring and satisfying the three important property of security. In the proposed architecture, CKMS plays a vital role and protect the file, which is shared among multiple users and protected from the intruder either from inside or outside. And also a small attempt is made to compare the different public key infrastructure algorithms in terms of time consumed to generate the key and time consumed during the process of encryption and decryption. Key translation protocol is used to share the key among owner, users, and CKMS, which will also ensure authenticity. Here, in the proposed architecture, we focused only on providing privacy for the file or the data stored in the cloud, but not on access level control. So the biggest open challenge is to handle efficiently the revocation mechanism, accountability, and user access control. Finally, we tested the metrics like time consumption for encryption/decryption and CPU and memory usage for the different PKI algorithms.

Data Availability

As a part of my research work, we used a multiple virtual machine within the private cloud and monitored the parameters like CPU time and its usage while generating the key of different sizes with respect to algorithms and also consumption of time during encryption and decryption of the proposed algorithm as mentioned in the paper. All the required metadata are also mentioned in the implementation section.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] P. Mell and T. Grance, *The NIST Definition of Cloud Computing, Version 15*, National Institute of Standards and Technology, Gaithersburg, MD, USA, 2009, http://csrc.nist.gov/groups/SNS/cloud-computing.

[2] G. Fowler and B. Worthen, "The Internet industry is on a cloud–whatever that may mean," *Wall Street Journal*, vol. 26, 2009.

[3] N. Leavitt, *Is Cloud Computing Really Ready for Prime Time?*, IEEE Computer Society Press, Los Alamitos, CA, USA, 2009.

[4] L. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds: towards a cloud definition," *Computer Communication Review*, vol. 39, no. 1, pp. 50–55, 2008.

[5] L. Youseff, M. Butrico, and D. Da Silva, "Toward a unified ontology of cloud computing," in *Proceedings of the 2008 Grid Computing Environments Workshop IEEE*, Austin, TX, USA, September 2008.

[6] A. Gholami and E. Laure, "Security and privacy of sensitive data in cloud computing: a survey of recent developments," *Computer Science and Information Technology*, vol. 5, 2016.

[7] R. P. Padhy, M. R. Patra, and S. C. Satapathy, "Cloud computing security issues and research challenges," *International Journal of Computer Science and Information Technology & Security*, vol. 1, no. 2, pp. 136–146, 2012.

[8] K. Manpreet and H. Singh, "A review of cloud computing security issues," *International Journal of Advances in Engineering and Technology*, vol. 8, no. 3, p. 397, 2015.

[9] H. Suo, Z. Liu, J. Wan, and K. Zhou, "Security and privacy in mobile cloud computing," in *Proceedings of 9th International Wireless Communications and Mobile Computing Conference (IWCMC)*, pp. 655–659, Sardinia, 2013.

[10] S. Nepal, C. Friedrich, Henry et al., "A secure storage service in the hybrid cloud," in *Proceedings of 2011 Fourth IEEE International Conference on Utility and Cloud Computing (UCC)*, New York, NY, USA, 2011.

[11] Y. Zhang, Q. Chen, and S. Zhong, "Privacy-preserving data aggregation in mobile phone sensing," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 5, pp. 980–992, 2016.

[12] H. Kwon, C. Hahn, D. Koo, and J. Hur, "Scalable and reliable key management for secure deduplication in cloud storage," in *Proceedings of the IEEE 10th International Conference on Cloud Computing (CLOUD)*, Honolulu, HI, USA, June 2017.

[13] K. V. Pradeep and V. Vijayakumar, "Survey on the key management for securing the cloud," *Procedia Computer Science*, vol. 50, 2015.

[14] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proceedings of Advances in Cryptology (Eurocrypt '99)*, pp. 223–238, Prague, Czech Republic, May 1999.

[15] Z. E. Dawahdeh, S. N. Yaakob, and A. M. Sagheer, "Modified ElGamal elliptic curve cryptosystem using hexadecimal representation," *Indian Journal of Science and Technology*, vol. 8, no. 15, pp. 1–8, 2015.

[16] A. R. Buchade and R. Ingle, "Key management for cloud data storage: methods and comparisons," 2014.

[17] H. Nayana and S. S. Manvi, "Thesis proposal summary: key management authentication and non repudiation for

information transaction in vehicular cloud environments," in *Proceedings of the 2016 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM) IEEE*, Bangalore, India, October 2016.

[18] O. Ethelbert, F. F. Moghaddam, P. Wieder, and R. Yahyapour, "A JSON token-based authentication and access management schema for Cloud SaaS applications," in *Proceedings of the 2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud) IEEE*, Prague, Czech Republic, August 2017.

[19] R. Chandramouli, S. Chokhani, and M. Iorga, "Cryptographic key management issues & challenges in cloud services," NIST Interagency or Internal Report 7956, 2013.

[20] H. Wang, Y. Zhang, H. Xiong, and B. Qin, "Cryptanalysis and improvements of an anonymous multi-receiver identity-based encryption scheme," *IET Information Security*, vol. 6, no. 1, pp. 20–27, 2012.

[21] Q. Wu, B. Qin, L. Zhang, J. Domingo-Ferrer, and J. A. Manjón, "Fast transmission to remote cooperative groups: a new key management paradigm," *IEEE/ACM Transactions on Networking*, vol. 21, pp. 621–633, 2013.

[22] S. Rajarajeswari and K. Somasundaram, "Data confidentiality and privacy in cloud computing," *Indian Journal of Science and Technology*, vol. 9, no. 4, pp. 1–8, 2016.

[23] R. Sugumar and S. B. S. Imam, "The symmetric encryption algorithm to secure outsourced data in public cloud storage," *Indian Journal of Science and Technology*, vol. 8, no. 23, pp. 1–5, 2015.

[24] R. Saikeerthana and A. Umamakeswari, "Secure data storage and data retrieval in cloud storage using cipher policy attribute based encryption," *Indian Journal of Science and Technology*, vol. 8, no. S9, pp. 318–325, 2015.

[25] L. Harn, C.-C. Chang, and H.-L. Wu, "An anonymous multi-receiver encryption based on RSA," *Journal of Network Security*, vol. 15, pp. 307–312, 2013.

[26] C. Alliance, "Security guidance for critical areas of focus in cloud computing v3.0," *Cloud Security Alliance*, vol. 15, 2011.

[27] J. Wayne and G. Timothy, *NIST Guidelines on Security and Privacy in Public Cloud Computing*, Draft Special Publication, NIST, Gaithersburg, MD, USA, 2011.

[28] S. Dorwani, J. Aghav, and B. Bhutkar, "NoSQL: features, classification, and comparison," in *Proceedings of the IEECS-2013*, Coimbatore, India, April 2016.

[29] L. Harn and C.-F. Hsu, "A Practical Hybrid Group Key Establishment for Secure Group Communications," *The Computer Journal*, vol. 60, no. 11, pp. 1582–1589, 2017.

[30] K. Bicakci, D. Deniz Yavuz, and S. Gurkan, "TwinCloud: secure cloud sharing without explicit key management," in *Proceedings of the 2016 IEEE Conference on Communications and Network Security (CNS)*, Philadelphia, PA, USA, October 2016.

[31] H. Dinh, A. Dworkin, C. O'Neill et al., "Omnibox: Efficient cloud storage by evaluating dropbox and box," in *Proceedings of the 2017 24th International Conference on Telecommunications (ICT) IEEE*, Limassol, Cyprus, May 2017.

[32] I. Indu, P. M. Rubesh Anand, and S. P. Shaji, "Secure file sharing mechanism and key management for the mobile cloud computing environment," *IJST*, vol. 9, 2016.

[33] G. Jindi, D. Zishan, and S. Lei, "Research on key management infrastructure in cloud computing environment," in *Proceedings of the International Conference on Grid and Cooperative Computing*, pp. 404–407, Nanjing, China, November 2010.

[34] A. Kumar, B. G. Lee, H. J. Lee et al., "Secure storage and access of data in cloud computing," in *Proceedings of the 2012 International Conference on ICT Convergence (IC-TC)*, pp. 336–339, New York, NY, USA, October 2012.

[35] F. H. Chen, Y. Liu, and Y. U. Qi, "Key management scheme of personal cloud computing," *Modern Computer*, vol. 30, 2011.

[36] S. D. C. D. Vimercati, S. Foresti, S. Jajodia et al., "Over-encryption: management of access control evolution on outsourced data," in *Proceedings of the 33rd International Conference on Very Large Data Bases*, Vienna, Austria, September 2007.

[37] A. Armando, R. Carbone, L. Compagna, J. Cuéllar, G. Pellegrino, and A. Sorniotti, "An authentication flaw in browser-based single sign-on protocols: impact and remediations," *Computers & Security*, vol. 33, pp. 41–58, 2013.

[38] D. Yan and X. Zhou, "Secure group communications using key hypergraphs," *Journal of Computational Information Systems*, vol. 8, pp. 5035–5042, 2012.

Hindawi
Journal of Computer Networks and Communications
Volume 2019, Article ID 7198435, 14 pages
https://doi.org/10.1155/2019/7198435

Research Article

A Case-Based Reasoning Approach for Automatic Adaptation of Classifiers in Mobile Phishing Detection

San Kyaw Zaw and Sangsuree Vasupongayya

Department of Computer Engineering, Faculty of Engineering, Prince of Songkla University, Hatyai, Songkhla 90112, Thailand

Correspondence should be addressed to Sangsuree Vasupongayya; vsangsur@coe.psu.ac.th

Received 5 April 2019; Accepted 28 May 2019; Published 19 June 2019

Guest Editor: Arash H. Lashkari

Currently, the smartphone contains lots of sensitive information. The increasing number of smartphone usage makes it more interesting for phishers. Existing phishing detection techniques are performed on their specific features with selected classifiers to get their best accuracy. An effective phishing detection approach is required to adapt the concept drift of mobile phishing and prevent degradation in accuracy. In this work, an adaptive phishing detection approach based on case-based reasoning technique is proposed to handle the concept drift challenge in phishing apps. Several experiments are conducted in order to demonstrate the design decision of our proposed model. The proposed model is evaluated with a large feature set containing 1,065 features from 10 different categories. These features are extracted from more than 10,000 android applications. Five combinations of features are created in order to mimic new real-world Android apps to evaluate our experiments. Moreover, a reduced feature set is also studied in this work in order to improve the efficiency of the proposed model. Both accuracy and efficiency of the proposed model are evaluated. The experimental results show that our proposed model achieves acceptable accuracy and efficiency for the phishing detection.

1. Introduction

Mobile communication is becoming more and more important within the context of Industry 4.0 [1]. The topmost security concern for mobile services is the phishing attack which can violence all confidential information of the mobile user [2]. Phishing attacks are increasing and evolving from a variety of newer methods despite the use of a number of detection approaches to battle mobile phishing attacks. Wombat Security revealed that 83% of organizations experienced phishing attacks in 2018 [3]. Figures published by the UK cyber security firm Alert Logic cited that phishing attacks, ransomware, and data loss as the top concerns [4]. Moreover, cybercrimes such as advanced persistent threats (APTs) and ransomware often start from phishing [5]. Currently, the phishers certainly try to hide their malicious payloads from a detection system using methods such as emulator detections, applications icon hiding, and reflection. APWG Phishing Attack Trends Reports released in March 2019 said a detection of phishing sites has become harder because phishers were obfuscating phishing URLs with multiple redirections [6]. In the context of machine learning, this phenomenon is known as concept drift and it becomes the main challenge to mobile phishing detections. Thus, the machine learning classifiers, applied in phishing detection models, must adapt to this concept drift in order to prevent any degradation in their detection accuracy.

In earlier phishing detection works, the variation of individual machine learning classification algorithm was applied. Each earlier phishing detection approaches showed an acceptable detection accuracy while using specific feature patterns with selected detection algorithms in their specific application domain [7, 8]. Currently, the usage of individual classification algorithm in phishing detection is developing to a combination of multiple classifiers in the form of ensemble methods to produce a better accuracy with more efficiency [9–11]. Unfortunately, most existing ensemble classification techniques in phishing detection could not afford to adapt automatically on the variation of input feature patterns, and it remains as a challenging issue in the phishing detection works [12]. Therefore, finding a way to make automatic adaptation classifiers based on the variation

of input features pattern will improve the key quality criteria of phishing detection, accuracy, and efficiency.

The main objective of this work is to create a mobile phishing detection system using a case-based reasoning approach for an automatic adaptation of classifiers according to the incoming feature patterns. By addressing the optimal selection of the suitable classifier to the incoming features using a case-based reasoning approach, the proposed mobile phishing detection system could provide the best performance by combining the good performance of all used methods appropriately. An adaptive phishing detection system based on a case-based reasoning (CBR) technique, which can handle the concept drift challenge in phishing apps, is proposed in this work. CBR is applied to construct a phishing detection model. A knowledge base or case base will control the detection algorithm by utilizing phishing features as cases. Moreover, an experimental analysis to verify that our proposed case-based phishing detection is suitable for handling concept drift of mobile phishing attacks than existing detection approaches will be conducted.

The rest of the paper is organized as follows. In Section 2, the background information of phishing attacks on smartphone is presented. The machine learning techniques on phishing detection and the background of case-based reasoning which models the ensemble of classifier approaches as cases in a knowledge base are also illustrated. Next, the overview architecture of the proposed adaptive phishing detection system and their detail processes are described in Section 3. The accuracy and performance analysis of the proposed system is presented in Section 4. The conclusions are described in Section 5.

2. Theoretical Background

The background technologies are described in this section. The nature of phishing attacks on smartphones and their attack techniques are presented in Section 2.1. Section 2.2 presents the literature review on existing phishing detection solutions based on machine learning techniques and their frequently used features. Lastly, case-based reasoning classification techniques are explained in Section 2.3.

2.1. Phishing on Smartphone. Nowadays, phishers are motivated to target smartphones due to several different reasons. A smartphone today is as powerful as a desktop or laptop computer. Smartphones usually contain lots of sensitive information of their owners. The increasing number of smartphone usage makes it more interesting for attackers. The phishing attack techniques are based on two categories including application-oriented phishing attacks and website-oriented phishing attacks.

The application-oriented phishing attacks can be categorized into two types based on their launching methods. First, the phishing application attempts to hijack (task interception) existing legitimate applications and continuously performs task polling. The phishing application will launch itself as long as it detects the launch of the target applications. These task interception attacks are specially based on the fake graphical user interface (GUI) techniques, which can easily impersonate and are hard to detect since a large touch screen is used as the primary user interface on most smartphones. As a result, the fake login interface is layered over the top of the real one, and the phishing app appears to be the target app. Second, the phishing application (repackaged applications) can directly present itself as the targeted legitimate app. This event may occur when the user downloads the fake applications from an unofficial app market. The website-oriented phishing attacks can also be categorized into two types based on their techniques. First, a phishing website hides (spoof) the URL bar of the targeted websites. Second, a phishing website attempts to overlay the genuine website with a crafted pop-up window. The spoofing URL is the process of creating a fake or a forged URL which impersonates a legitimate and secure website. This kind of URL spoofing attack is harmful and dangerous because the website looks exactly like the original one [13]. The fake website asks the user to enter his/her username, password, credit card number, or other information. For a legitimate mobile app that includes an embedded web page served over HTTP or a legitimate mobile app that allows the overlaying pop-up window, the network attacker can change the login button on the page or substitute a crafted pop-up window so that there is a link to a page owned by the attacker. When the user clicks the button, the user will be taken to the phishing page within the embedded web frame. This way, the attacker can steal the user credentials. The attacker can then relay the credentials to the valid website in order to mimic the normal work flow.

Existing solutions in phishing detection show an acceptable accuracy in their specific domain using their targeted features and their specified machine learning techniques. Thus, an effective phishing detection that is less dependent on the features pattern is still needed in this age. This work aims to propose an adaptive phishing detection by combining many existing techniques.

2.2. Current Phishing Detection Solutions. Existing phishing countermeasures use techniques such as content filtering, visual matching, and blacklist or whitelist matching [14]. Content filtering system examines the content of webpages for suspected URLs. Content filtering can be achieved by identifying statistical differences between legitimate and suspected phishing contents or constructing a set of rules [15, 16]. Visual matching computes a visual similarity between the phishing and the legitimate pages, based on the images, blocks, and layout [17, 18]. For a blacklisting system, the known phishing URLs are listed based on a human verification method. The very low false positive rates will be resulted in this approach. For a whitelisting system, users specify the links of trusted sites and frequently accessed websites. By contrary, other new websites will be suspected as phishing attacks [15].

Possible phishing attacks on mobiles which can launch during the control transfers are discussed in [19]. An indicator for the applications identity upon the navigation bar of the system to show the currently running application or

the current web page was implemented in [20]. The personalized security indicators to mobile apps are proposed in [21]. However, the user-driven decision-making process is still needed.

The unified and trusted login user interface is used in another group of antiphishing techniques. A software keyboard which can be used safely for login input is provided in [22]. For the purpose of handling the credential, the hardware and software certificates that are used to confirm the login is proposed in [23]. However, these approaches require some modifications to the client application and the user effort. An antiphishing system for mobile platforms was presented in [24]. The work was continued in [4] to detect the persistent account registry phishing attacks. They used OCR technique, and their database needs to save every snapshot of the protected applications and webpages. Using the QR code in phishing attacks was demonstrated and analyzed in [25]. They combined the client-server architecture with a digital signature to perform an integrity checking and authentication. However, the work only focused on the QR code phishing attacks while the phishing malware was not considered. Phishing Detective [26] was created to identify whether or not a link in the user e-mail might send the user to a phishing page. However, the work was totally relied on the blacklist URL of Phish Tank database; it might not be able to satisfy other types of phishing attacks such as activity hijacking and repackaging attacks.

MP-Shield [27] is an Android application that aims to inspect the flow of IP packets between the origin and the destination of mobile user applications. Their work mainly emphasized on the monitoring URL for detection purposes. The types of phishing attack that can be mounted on mobile devices were identified in [19]. The authors conducted an analysis of ways in which the mobile applications and the web sites link to each other. The common control transfer on mobile and how phishing attacks can be mounted against the control transfer scenarios were discussed. The authors presented possible types of phishing attacks along with their legitimate behaviors as summarized in Table 1.

According to Table 1, the mobile sender means a mobile application that sends the user to a website or another mobile application, while the web sender means a website that sends the user to a mobile application or other web sites.

Our work will cover these attack models with ten groups of selected feature categories. Each phishing detection approach showed an acceptable detection accuracy while using different features. Unfortunately, majority of phishing detections may suffer the lack of features for efficient detection of phishing malwares. An optimized solution which used different kinds of features of Android applications to prevent the phishing and malware on Android smartphone is still needed. Our work will contribute to the finding of an optimal solution for mobile phishing detection in the sense of using the feature independently with various classifiers.

2.3. Case-Based Reasoning. Case-based reasoning (CBR) is a problem-solving approach that solves new problems by adapting or reusing old solutions that were used to solve similar problems [28]. The past experience or previous problems are saved as cases, and each case contains representative features, characteristics of the problem, and its solution. The case base is a collection of these cases. The knowledge base of the problem-solving experience is used for the new problem solving [29]. The solutions in the retrieved cases are reused as a proposed solution to the new problem. Thus, the solution to the new problem can be found from similar known solution in the past.

If the new problem situation is exactly the same as the previous cases, then the reuse is simple. CBR systems start their reasoning from the knowledge unit, called cases, while the data-mining systems most often start from the raw data. CBR systems also belong to the instance-based learning systems in the field of machine learning, that are defined as systems that are capable of automatically improving their performance over time. As long as the CBR systems learn new cases in the retain step, they are qualified as the learning systems, thus belonging to the machine learning system [30]. The learning process of a case-based reasoning approach is shown in Figure 1.

Case-based reasoning system performs the learning process as follows:

(1) Retrieving the most similar case or cases from the case base to the new problem
(2) Reusing the previous solutions of the similar cases to solve the new problem
(3) Revising the proposed solution (if necessary)
(4) Retaining the solution of the new case for future problem solving

A new problem to the system is represented as a case and is compared with existing cases in the case base. The most similar case or cases are retrieved based on the similarity comparison of case representations. These retrieved cases are adapted (i.e., combined and reused) to propose a solution for the new problem. The suggested solution may need to be evaluated and corrected (i.e., revised) in some cases if it is not the best solution. This verified solution can be added back as a new case to the case base (i.e., retained) or as amendments to existing cases in the case base to be used in future problem solving [28].

3. Architecture Overview

A case-based reasoning model is proposed as an automatic adaptation of classifiers for mobile phishing detection. The information on how to design the case-based adaptive classification system is presented in this section. The proposed system consists of two main parts, including the application on Android smartphones and the detection system on the cloud environment. Figure 2 shows the overall system design.

As shown in Figure 2, the feature will be extracted from the Android application for the phishing detection process. The detailed information of features will be discussed in Section 3.1. Then, the extracted features will be sent to the cloud environment for phishing detection processes. As the

Table 1: Legitimate behaviors and their respective phishing attack techniques.

	Legitimate behavior	Respective attack techniques
Mobile sender	Social sharing, upgrades, game credits, opening a target in the browser, send user to embedded http page in browser that links to https login	Fake mobile login screen, task interception, scheme squatting, keylogging, URL bar hiding/spoofing, fake browser, using active network attack plus URL bar spoofing.
Web sender	Link to mobile e-mail or Twitter, payment via PayPal or Google checkout and user follows link from http to https	Website spoofs mobile app, task interception, scheme squatting, URL bar hiding/spoofing, active network attack plus URL bar spoofing.

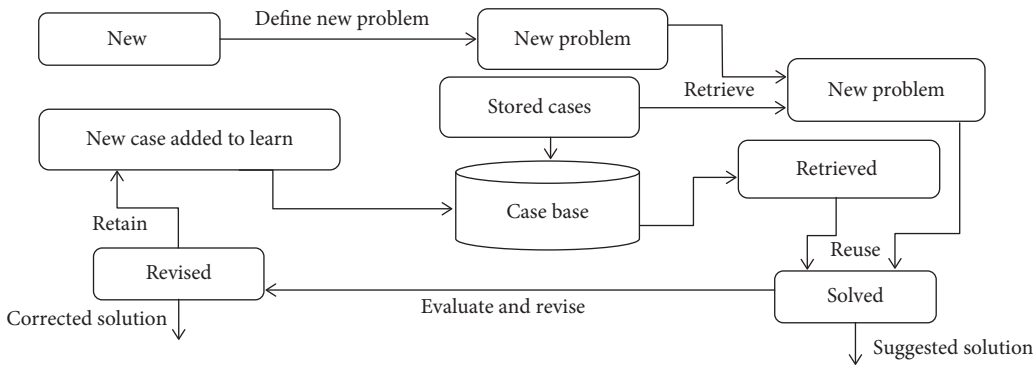


Figure 1: Case-based reasoning approach.

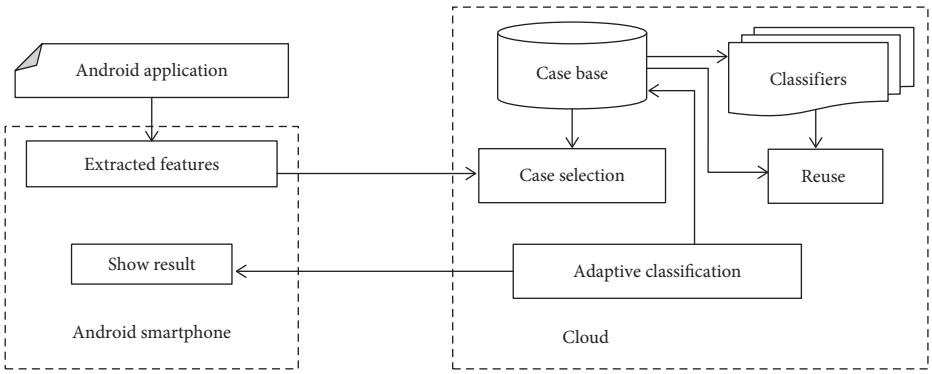


Figure 2: Overall system design.

main objective of this work is to enhance the phishing detection processes, the detection will be performed on the static and dynamic feature from Android malware dataset (described in Section 4.1). The detailed process of feature extraction is out of this paper scope.

The contribution of our work starts on the receiving of the extracted features by the detection system. The first process is to retrieve the most similar case from the case base (which stored previous Android phishing detection approach along with the corresponding features). The case-retrieving process will be described in Section 3.3. The case base must be set up before the case-retrieving process. The case base setting up process is shown in Figure 3. The details of the case base setting up process are presented in the following section.

According to the retrieved case, the most suitable classification techniques will be used for the adaptive classification. If the feature set extracted from the Android application does not match the sets of features stored in the case base, the adaptive classification will select the suitable methods to process the extracted feature set according to the similarity ratio score. The selection of suitable methods means choosing the multiple classifiers for the extracted feature set. Finally, the final result of the active Android application will be sent to the application on Android smartphone to be displayed to the user.

3.1. Feature for Mobile Phishing. Existing antiphishing solutions on mobile environments were collected, and their features were extracted to identify a phishing attack. Under an Android environment, the features can be extracted from miscellaneous sources such as program entities and program outputs of the runtime monitoring. The list of frequently used features by existing antiphishing solutions can be classified into ten classes including Android components, Android API counts, API usage action, security-sensitive

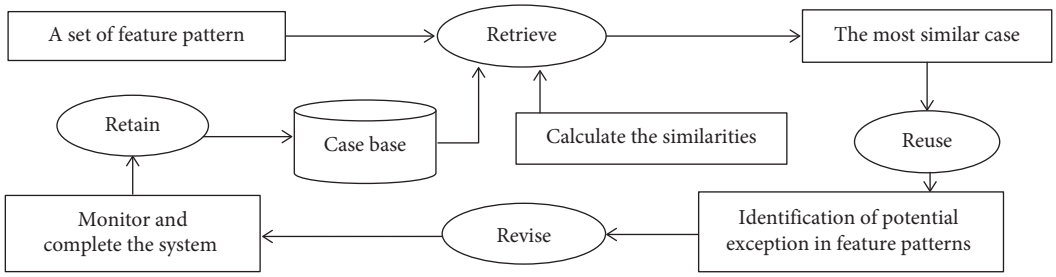


Figure 3: Setting up case base.

data flow, hardware components, intent actions, permissions, shell command and strings, contents, and visual, and URLs. The details of each feature are given below.

(1) Android components: a variety of component types with specific functionalities (e.g., components for providing GUIs, and others for running background services) are declared within an Android app's manifest, and these features are collected in [31–33].

(2) API count: the number of invocations of a specific Android API method (e.g., the malicious apps access the location APIs twice and the telephony package 8 times) are collected in [4, 24, 27, 32].

(3) API usage actions: APIs can be used to develop applications in Android platform and also misused by malicious purposes. There are many approaches to submit the web requests and to ex-filtrate the captured data via the API without the Internet permission. Some existing phishing detection works [27, 31, 32, 34] collect the API calls (e.g., API calls to access the sensitive data; API calls to access the network communications; API calls to send and receive the SMS messages; API calls to execute the external commands, and API calls frequently used for obfuscation).

(4) Security-sensitive data flows: a few approaches for Android malware detection [31, 34, 35] use data flows between security-sensitive Android interfaces to determine if an app is malicious. Tracking this form of information is particularly useful for identifying privacy leaks.

(5) Hardware components: the hardware components are listed in AndroidManifest.xml that is used in the app (e.g., to access the camera, an app needs to include android.hardware.camera feature) and these features are collected in [4, 36].

(6) Intent actions: Android malwares are known to rely upon tracking of an Intent (e.g., whether a package is installed, or if a device has recently completed booting) to determine when to perform a malicious behavior. These features are used in [32, 36].

(7) Permission: specific permissions, provided by Android to execute some risky operations, are acquired by Android malwares. These features are collected in [34, 37, 38].

(8) Shell command and strings: the features of interested strings associated with malicious behaviors and potential risky shell commands are collected in [36, 39]. Some of the structural attributes of APK file, such as size of code, presence of zip file, binary file, and related information, are also included in this feature group.

(9) Contents and visual: the main display channel for the deception of phishing is the web content which expresses the intention of the website. These features consist of the page elements, such as the page title, the submitted form, and the contained links. Some researchers also extract the logo icon and the contained pictures from the web page and use an image recognition algorithm to identify the phishing website [16–18].

(10) URLs: web link features for phishing fraud is collected based on five criteria, including URL and Domain Identity, Security and Encryption, Source Code and Java script, Page Style and Contents, and Web Address Bar. These features are collected in [4, 13, 40].

3.2. Case Representation. A case represents an experience at an operational level. Typically, a case includes the problem specification, the solution, and sometimes the outcome. This is the most common representation used. However, more elaborate case representations can be employed. Depending on the information included in a case, different types of results can be achieved from the system. Cases that describe a problem and its solution can be used to derive solutions to new problems.

In general, a case specification is described as a set of features. The features are those aspects of the domain and the problem that are considered to be most significant in determining the solution and/or outcome. A case represents an experience. In this situation, a case should represent the features of the application that is used to determine a phishing attack.

In our model, a case includes the combination of feature sets, ensemble method of classifiers, or individual classification algorithm with their specific parameters, the accuracy and performance of the solution, and potential facilitations. A case description stored in the phishing detection system is shown in Table 2.

Table 2: Case description for mobile phishing detection system.

No.	Name	Value
1	Case ID	Case identification number
2	Feature pattern	Combination of feature sets
3	Ensemble methods of classifiers (or) classification algorithm	Boosting/bagging/bayesian (or) algorithm name and their specific parameters
4	Accuracy	Percentage of correct classification
5	Performance	Runtime (seconds)

To define a new case in case base, the input features have to pass through different machine learning classifiers, and the results from each classifiers are calculated to produce the final result. Then, the input features, the classifiers with parameters, the activation function, and the final result are stored in the case base (knowledge base) as a new case. The process of defining a new case to be stored in the case base is shown in Figure 4.

3.3. Case Retrieval. Case-based reasoning (CBR) solves a new problem by retrieving the previously solved problems and their solutions from a knowledge source of cases, called the case base. There are challenges related to retrieving process that still need to be addressed. One issue is the computation of similarity, which is particularly important during the retrieving process. The effectiveness of a similarity measurement is determined by the usefulness of a retrieved case in solving a new problem.

The aim of using the CBR approach is the selection of the most similar past phishing detection cases to the new problem. A set of similar cases is selected from the case base according to a similarity criterion that requires the specification of weights corresponding to attributes. The assessment of case similarity involves the comparison of attribute values of the new case and that of the past cases, stored in the case base. The retrieved old cases are ranked according to their similarity scores to the attributes of the new case. In this work, the nearest neighbor method is applied to calculate the similarity score and the total similarity score of a potentially useful case.

3.4. Adaptive Classification System Design. The main objective of case-based adaptive classification is to assign a suitable classification technique to the target case (a feature set extracted from Android application) by identifying and analysing the training case (sets of features that are stored in the case base) that is similar. The proposed case-based adaptive classification is shown in Figure 5. If the feature set extracted from the active Android application do not match with any set of features, stored in the case base (that means the extracted feature set is not complete for the case-retrieving process), the adaptive classification will select suitable methods to process the extracted feature set. The selection of suitable methods has two options. First, the possible features are added to the extracted feature set in order to perform the case-retrieving process and to choose a suitable classifier. Second, multiple classifiers are selected to process the extracted incomplete feature set. Under the second option, multiple answers, resulted from multiple classifiers, are collected in order to produce a final answer by the way of weighted sum of all answers.

4. Detection Model and Evaluation

This section explains how our detection model performs adaptively on the combination of individual classifiers and ensemble classifier. To verify that our proposed model can improve the accuracy of the mobile phishing detection, an experiment is conducted using the feature sets (which has been described in Section 3.1). The experiment was conducted by running Weka 3.8 on a Laptop computer with core i7 processor, 8 GB RAM, and Windows8.1 64 bit operating system. The cross-validation method is used as an evaluation technique to estimate the error rate efficiently and in an unbiased way by running repeated percentage splits. Firstly, the dataset is divided into 10 pieces. Each piece is used as a testing dataset in turn while the remaining 9 pieces together are used as a training dataset. We preformed 10 simulations (i.e., experiments are repeated 10 times). Then, all these results are averaged as a single estimation result. Six of the existing machine learning algorithms are chosen from different categories and used with 10-fold cross-validation methods to evaluate the variation of accuracy and efficiency.

4.1. Dataset. The features are extracted from more than 10,000 Android malware samples which are collected from Android malware repositories including VirusShare [41], AndroZoo [42], Droid screening [43] and Reveal droid [44]. There are 76 extracted features of Android components, including 31 features of API counts, 82 features of API usage actions, 421 features of security-sensitive flows, 6 features of hardware components, 109 features of intents, 82 features of permissions, 190 features of malicious shell command and strings, 19 features of content visual, and 49 features of URLs. Thus, there are 1,065 features in total. The information of the feature sets used in this experiment is shown in Table 3.

4.2. Machine Learning Classifiers. To detect and classify the phishing applications, different machine learning classification techniques are used with an adaptive method. An adaptive classification system is proposed to automatically choose a combination of suitable classifiers for the extracted features of an active Android application. Various machine learning techniques were used as the classifier in existing works [31, 32, 34, 35]. Among them, six algorithms were selected from different categories for the coverage usage of all classification nature. The six algorithms include C4.5 (J48), decision table (DT), k-nearest neighbors (IBK), logistic regression (LR), naive Bayes (NB), and support vector machine (SVM). According to the pretesting on the effectiveness of parameter on these classifiers [45], naive Bayes (NB) classifier with supervised discretization function, the

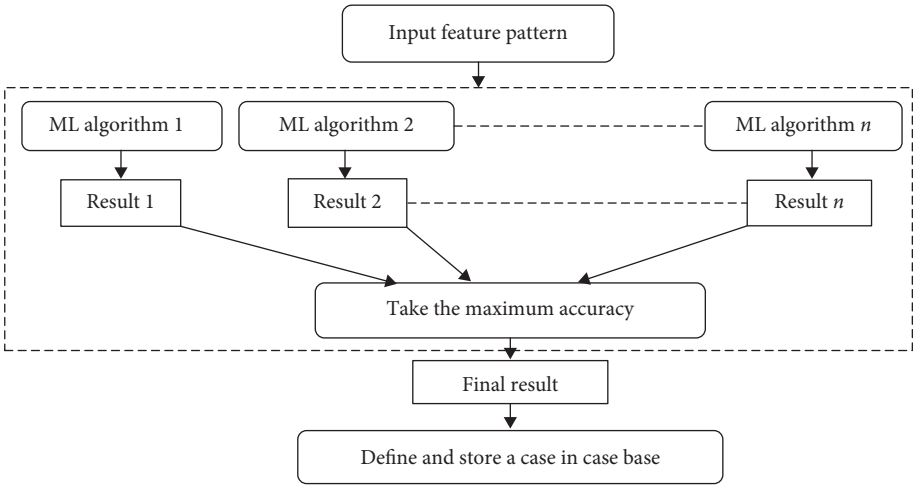


Figure 4: Case defining process (define a new case and store in case base).

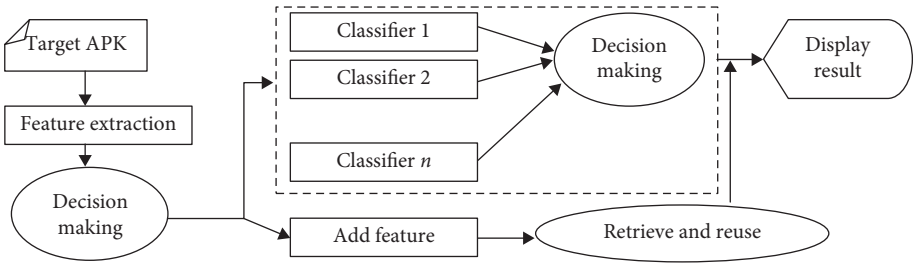


Figure 5: Adaptive classification.

Table 3: Feature sets.

No	Feature sets	Number of features	Example features
1	Android components	76	android.media, android.media.effect, android.media.audiofx, android.service.textservice, android.service.notification
2	API counts	31	account_information, account_settings, audio, bluetooth, bluetooth_information
3	API usage actions	82	android.util, android.widget, android.renderscript, android.webkit, android.os, android.os.storage, android.content
4	Security-sensitive flows	421	system_settings____audio, system_settings____phone_connection, system_settings____voip, system_settings____database_information
5	Hardware components	6	android.hardware.display, android.hardware, android.hardware.usb, android.hardware.location, android.hardware.input
6	Intent_action	109	action_main, action_view, action_default, action_attach_data, action_edit, action_insert_or_edit
7	Permission	82	android.permission.access_cache_filesystem, android.permission.access_checkin_properties, android.permission.access_coarse_location, android.permission.access_gps
8	Shell_command_strings	190	runtime.exec, createSubprocess, cipher-classes, longstring, SecretKey, method.invoke, small_code_size
9	Content_visual	19	HostnameLength, PathLength, QueryLength, DoubleSlashInPath, NumSensitiveWords, EmbeddedBrandName, PctExtHyperlinks,
10	URLs	49	having_ip_address, url_length, shortining_service, having_at_symbol, double_slash_redirecting, prefix_suffix,
	Total	1,065	

default maximum number of iterations in logistic regression (LR), the confidence factor of 0.5 for pruning tree for J48 classifier, and a 1-nearest neighbors (IBK) classifier are chosen for our experiment. SVM and decision table classifiers are used with their default parameters.

4.3. Experimental Results and Analysis. The accuracy comparison of six classifiers on the 10 feature sets is shown in Table 4. The italicized values shown in Table 4 represent the maximum detection accuracy among six classifiers for each feature set. It can be seen that the accuracy of each

Table 4: Accuracy comparison of classifiers on 10 features.

	Feature sets	J48 (%)	DT (%)	IBK (%)	LR (%)	NB (%)	SVM (%)
1	Android components	93.23	89.02	*93.40*	90.16	84.67	87.95
2	API count	*95.85*	93.02	95.66	91.90	89.20	85.25
3	APIusage_actions	95.20	91.86	*95.32*	91.97	89.02	91.24
4	Flow	93.05	91.03	*93.32*	87.18	87.45	83.17
5	Hardware components	89.00	89.06	*89.12*	89.06	89.02	89.06
6	Intent_action	86.89	85.73	*87.13*	84.64	83.75	85.53
7	Permission	94.30	91.92	*94.65*	93.95	88.54	94.14
8	Shell_command_strings	*75.40*	71.18	74.08	70.28	68.74	70.22
9	Content_visual	*97.20*	95.79	95.53	94.49	95.77	93.87
10	URLs	*96.03*	93.24	97.18	93.99	92.98	93.80

classification algorithm depends on the features. IBK can provide a better accuracy in 6 features, and J48 can provide a better accuracy in other 4 features. Our work aims to detect mobile phishing in the nature of feature independent with various classifiers. To create a real-world application, a random feature combination is created because a new Android application can consist of any combination of features. In this experiment, 5 random combinations of features are created, as shown in Table 5.

These 5 feature combination patterns are tested with individual six classifiers and three models of ensemble classifiers to develop a case for our adaptive model. Each model is an ensemble of six classifiers with different methods in providing the final answer. The final answer finding methods of ensemble classifiers include the average of probabilities, majority voting, and maximum probabilities. The detection results for 5 scenarios of random feature combination sets with the six base classifiers and three ensemble classifiers are described in Table 6. The italicized values shown in Table 6 represent the maximum detection accuracy of 5 cases among nine classifiers.

According to the results shown in Table 6, some feature patterns are more suitable with ensemble techniques while some are better used with individual classification techniques. It can conclude that the accuracy variation of classification techniques in mobile phishing detection heavily relies on the input features.

The adaptive method used in our model will choose the most suitable classification approach for a set of input features. Based on the results presented in Table 6, we can develop a case to be stored in case base for an adaptive choice of suitable classifiers. The tentative cases for building our case-based phishing detection model is shown in Table 7.

Performing the classification process on these large numbers of features takes a long runtime. The comparison of runtime to build the detection model on 6 base classifiers and 3 ensemble approaches before selecting the feature is shown in Table 8.

To reduce the detection time, some features may be omitted because the features may not provide a high impact on the result. Therefore, some experiments are conducted to select a set of effective features in order to reduce the number of required features.

4.4. Selecting the Features. Feature selection is necessary to reduce the dimension of the feature space. With the aim of getting the benefits of performing a feature selection technique on a large data set such as reducing an overfitting issue, improving accuracy, and reducing a processing time, two feature selection techniques are performed in this experiment by comparing their results to get the optimized results. The process of selecting the features can be described by the following steps.

Let U be the universe of feature sets $U = \{D_1, D_2, \ldots, D_v\}$. And, the dataset $D_i \in U$, with number v of attributes A, be $D_i = \{A_1, A_2, \ldots, A_v\}$. Then, the attributes can be grouped into feature group FG_i as $\mathrm{FG}_i = \{A_a, A_b, \ldots, A_n\}$. Some attribute evaluation is performed and selected on the worth of each attribute, which becomes a selected feature set $\mathrm{FS}_i = \{A_a, A_b, \ldots, A_m\}$, where $\mathrm{FS}_i \in \mathrm{FG}_i$.

Two methods of feature selection techniques are used in this experiment to confirm the advantages of selecting the features in phishing detections. The first method is a correlation-based feature selection with a ranker search method that evaluates each attribute and lists the results in a ranked order. The worth of each attribute is evaluated by measuring the correlation (Pearson's) between it and the class [46].

Pearson's correlation coefficient is described in equation (1), where all variables have been standardized. The correlation between a composite and a class label is a function of the number of component variables (attributes) in the composite and the magnitude of the intercorrelations among them, together with the magnitude of the correlations between the attributes and the class label.

If the correlation between each of the attributes in a test and the class label is known, and the intercorrelation between each pair of attributes is given, then the correlation between a composite test consisting of the summed attributes and the class label can be predicted from the following equation:

$$r_{zc} = \frac{k\overline{r_{zi}}}{\sqrt{k + k(k-1)\overline{r_{ii}}}}, \tag{1}$$

where r_{zc} is the correlation between the summed attributes and the class label; k is the number of attributes; r_{zi} is the average of the correlations between the attributes and the class label; and r_{ii} is the average intercorrelation between attributes.

We get the ranked attributes listed with their corresponding class correlation. Some attributes, which owned no

Table 5: Scenarios for random combinations of features.

Case ID	Feature pattern	Combination of feature sets	Number of features
01	Pattern 1	API count + API usage + hardware	112
02	Pattern 2	API count + intent	139
03	Pattern 3	API count + API usage + intent + hardware	220
04	Pattern 4	Flow + intent	529
05	Pattern 5	Flow + intent + API usage + hardware	610

Table 6: Detection accuracy of 5 scenarios on randomly combined feature patterns.

Case ID	J48 (%)	DT (%)	IBK (%)	LR (%)	NB (%)	SVM (%)	AVG (%)	MAJ (%)	MAX (%)
01	*95.93*	93.07	95.45	92.47	89.42	91.62	95.31	95.31	92.87
02	*94.72*	91.62	94.04	90.18	86.44	89.27	94.26	94.20	91.38
03	96.32	92.67	95.60	94.89	90.69	92.57	*96.43*	96.41	94.31
04	90.56	86.38	90.45	88.51	81.55	87.88	*90.64*	*90.64*	88.52
05	95.33	89.69	94.37	93.97	92.28	91.61	95.68	*95.69*	92.68

Table 7: Tentative cases for mobile phishing detection system.

Case ID	Feature pattern	Adaptive method	Accuracy (%)	Run time (seconds)
1	Pattern 1	J48	95.93	4.43
2	Pattern 2	J48	94.72	4.54
3	Pattern 3	AVG	96.43	95.18
4	Pattern 4	AVG, MAJ	90.64	174.4, & 174.6
5	Pattern 5	MAJ	95.69	205.50

or less values on the class correlation measures, are eliminated. The resulting reduced feature sets are shown in Table 9.

The second method is an information gain attribute evaluation-based feature selection with a ranker search method. Information gain ratio evaluation is calculated by using the following equations. In the attribute evaluation processes, I index measures the impurity of D; a data partition or a set of training tuples is calculated using

$$I(D) = 1 - \sum_{i=1}^{m} p_i^2, \tag{2}$$

where p_i is the probability that a tuple in D belongs to class C_i and is estimated by $(|C_{i,D}|/|D|)$. The sum is computed over m classes when I index considers a binary split for each attribute. First, the case where A is a discrete-valued attribute having v distinct values, $\{A_1, A_2, \ldots, A_v\}$ occurring in D is considered. The expected information provided by that split is calculated by

$$I_A(D) = \sum_{i=1}^{v} \frac{|D_j|}{|D|} \times I(D_j). \tag{3}$$

In this equation, D_j represents the observations that contain the jth attribute. The information gain of a binary split on attribute A is calculated by

$$\text{Gain}(A) = I(D) - I_A(D). \tag{4}$$

Information gain ratio attempts to correct the information gain calculation by introducing a split information value. The mathematical formulation for split information is provided in

$$\text{SplitInfo}_A(D) = -\sum_{i=1}^{v} \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right). \tag{5}$$

This value represents the potential information generated by splitting the training dataset, D, into v partitions, corresponding to the v outcomes of a test on attribute A. The gain ratio is defined in

$$\text{Gain ratio } (A) = \frac{\text{Gain}(A)}{\text{SplitInfo}_A(D)}. \tag{6}$$

The attribute with the maximum gain ratio is selected as the highest ranked attribute. The low-ranked attributes that provide a gain ratio less than 0.0003 are eliminated. After performing the two feature selection techniques on the dataset, the reduced feature sets are generated as shown in Table 9.

The same detection experiments are conducted with 9 classifications on each selected feature set. The detection results of 5 cases on selected feature sets are described in Tables 10 and 11. In this experiment, 9 classification approaches with their related parameters are set up as the same as that of previous experiments (described in Section 4.2).

According to the results of the reduced datasets with a correlation attribute evaluation method shown in Table 10, the classification approaches with the best detection accuracy are slightly changed in 2 cases (feature patterns 3 and 4). Feature pattern 3 is a combination of API count, API usage, Intent, and Hardware. The italicized values shown in Table 10 represent the maximum detection accuracy of 5 cases among nine classifiers. The highest detection accuracy is now provided by ensembles with AVG and MAJ final answer methods, while the highest detection accuracy is provided by ensembles with the AVG final answer method when full feature set is used. The detection accuracy is slightly increased for most classifiers in feature pattern 4 which is a combination of flows and Intents features.

According to the results shown in Table 11 of the reduced datasets with an information gain attribute evaluation

Table 8: Runtime comparison of 5 scenarios on 9 classification approaches (in seconds).

Case ID	J48	DT	IBK	LR	NB	SVM	AVG	MAJ	MAX
01	4.43	18.63	0.01	1.93	0.64	3.01	29.87	29.52	26.16
02	4.54	31.80	0.0001	2.57	0.66	39.24	76.20	76.35	75.90
03	9.44	58.80	0.01	7.22	1.14	18.94	95.18	96.09	97.41
04	12.09	148.32	0.0001	5.28	1.39	6.25	174.4	174.6	174.61
05	17.09	167.14	0.01	7.86	1.93	3.61	203.62	205.50	203.51

Table 9: Information of selected feature sets for 5 cases.

Case ID	Feature combination pattern	Features before feature selection	Features selected by Pearson's correlation	Features selected by information gain
01	Pattern 1	112	96	100
02	Pattern 2	139	114	120
03	Pattern 3	220	180	185
04	Pattern 4	529	164	265
05	Pattern 5	610	227	250

Table 10: Detection accuracy of 5 cases after correlation attribute evaluation feature selection.

Case ID	J48	DT	IBK	LR	NB	SVM	AVG	MAJ	MAX
01	*95.87*	93.10	95.45	92.47	89.42	91.61	95.31	95.32	92.82
02	*94.68*	91.53	94.04	90.18	86.44	89.28	94.24	94.18	91.33
03	96.37	92.70	95.60	94.90	90.69	92.57	*96.38*	*96.38*	94.37
04	*90.73*	86.51	90.45	88.51	81.55	87.89	*90.73*	90.72	88.64
05	95.38	89.54	94.37	93.96	92.28	91.61	95.68	*95.69*	92.72

Table 11: Detection accuracy of 5 cases after information gain attribute evaluation feature selection.

Case ID	J48	DT	IBK	LR	NB	SVM	AVG	MAJ	MAX
01	*95.96*	93.10	95.62	92.54	89.42	91.66	95.37	95.37	92.83
02	*94.66*	91.53	94.16	90.15	86.44	89.19	94.19	94.12	91.30
03	96.38	92.59	95.55	95.02	90.69	92.61	*96.45*	*96.45*	94.34
04	90.52	86.36	90.24	88.70	81.55	87.86	*90.77*	90.76	88.69
05	95.46	89.44	94.50	93.98	92.28	91.56	*95.80*	95.79	92.82

method, the detection accuracy is increased in 4 cases (feature patterns 1, 3, 4, and 5). The italicized values shown in Table 11 represent the maximum detection accuracy of 5 cases among nine classifiers. Moreover, the classification approaches which produced the best detection accuracy are changed in 3 cases (feature patterns 3, 4, and 5). That is, an ensemble with AVG final answer finding method provides the best accuracy for feature patterns 3, 4, and 5.

The detection accuracy percentages of 5 cases by using different algorithms are comparatively described in Figure 6. This figure represented the detection results from Tables 6, 10, and 11. Each case is represented in 3 situations such as no features selection, after correlation attribute evaluation feature selection, and after information gain attributes evaluation feature selection. There are 15 points in the figure, representing the 5 cases with 3 conditions. The best classifier for case 01 and case 02 is J48 classifier while ensemble classifier AVG is the best one for case 03, case 04, and case 05. The cases with the best algorithm are used in the case-based reasoning detection method.

With the aim of highlighting the performance of feature selection techniques, the runtime results of reduced feature sets are collected as described in Tables 12 and 13. The information gain attribute evaluation method results in a large number of features than the correlation attribute evaluation method. The runtime of the information gain attribute evolution method is also slightly larger than that of the correlation attribute evaluation method.

The runtime on 5 cases by selecting the features are showed in Figure 7. This figure compared the runtime from Tables 8, 12, and 13. There are 15 points in the figure representing the 5 cases with 3 conditions.

Selecting the features with the information gain attribute evaluation approach is applied on our feature sets to improve our model for better accuracy and efficiency. The percentages of detection accuracy on 4 feature patterns are improved as shown in Table 11 while the performances of the detection on all feature patterns are improved as shown in Table 13. Table 14 shows the comparison of accuracy and efficiency of full feature sets and reduced feature sets of our proposed adaptive model. The italicized values shown in Table 14 represent the accuracy values when a reduced feature set is used, and the accuracy values are improved over their counterpart when a full feature set is used.

The phishing malware detection task is an imbalanced classification problem. That is, there are two classes to be

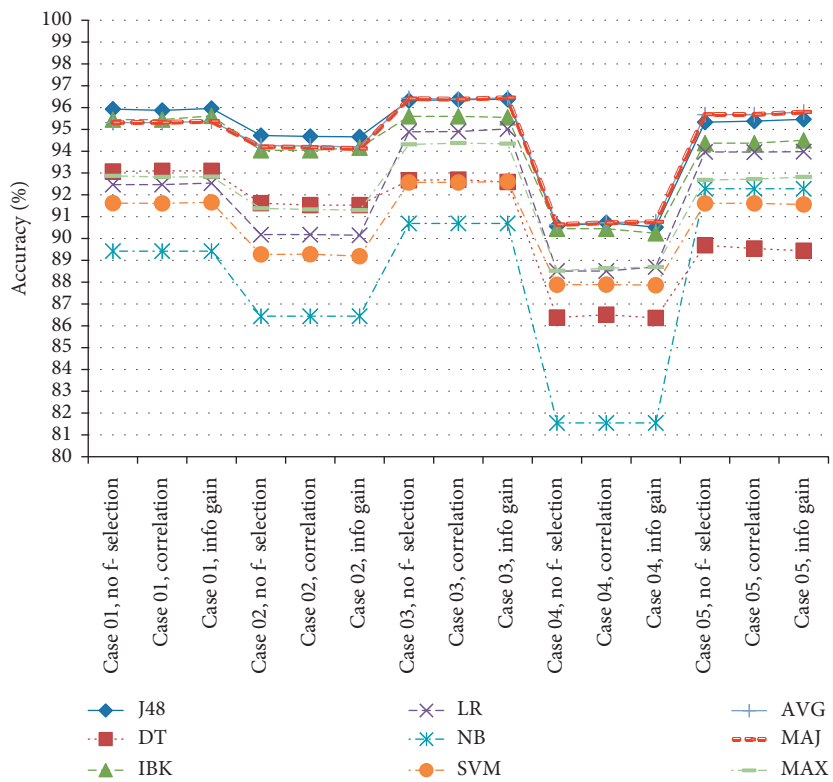


Figure 6: Accuracy comparison of 9 classifiers on 5 cases before and after feature selection.

Table 12: Runtime comparison after correlation attribute evaluation feature selection (seconds).

Case ID	J48	DT	IBK	LR	NB	SVM	AVG	MAJ	MAX
01	3.94	18.36	0.01	1.97	0.64	3.38	27.45	26.92	27.18
02	3.84	25.35	0.0001	2.43	0.50	39.46	72.07	72.12	71.19
03	8.06	45.85	0.01	7.20	1.03	19.55	83.10	83.52	83.34
04	5.60	44.75	0.0001	5.15	0.56	6.27	61.95	61.99	62.00
05	8.84	69.88	0.0001	7.65	1.00	3.20	90.23	90.06	90.45

Table 13: Runtime comparison after information gain attribute evaluation feature selection (seconds).

Case ID	J48	DT	IBK	LR	NB	SVM	AVG	MAJ	MAX
01	4.05	20.25	0.01	1.63	0.54	3.02	29.02	27.49	27.27
02	3.86	29.43	0.001	2.35	0.56	31.96	67.63	66.52	66.40
03	9.77	55.46	0.01	6.31	0.95	17.06	87.31	90.63	90.69
04	6.83	87.36	0.01	2.25	0.93	6.76	102.86	93.21	93.15
05	8.42	104.52	0.001	5.37	1.21	3.95	111.80	107.53	108.07

identified, including phishing and benign, with one category representing the overwhelming majority of the data points. In these cases, the positive class "phishing" is greatly outnumbered by the negative class. These types of problems are examples of the fairly common case in the data science when the accuracy is not a good measure for assessing the model performance. Intuitively, proclaiming all data points as negative in the phishing detection problem is not helpful and, instead, we should focus on identifying the positive cases.

In order to assess the effectiveness of our proposed model, the confusion matrix evaluation is applied: accuracy, precision, and sensitivity. While sensitivity expresses the ability of a model to find all relevant instances in the dataset, precision expresses the proportion of the instances that our model predicts as positive and they are actually positive. The following formulas represent their definitions:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}},$$
$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{7}$$
$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}}.$$

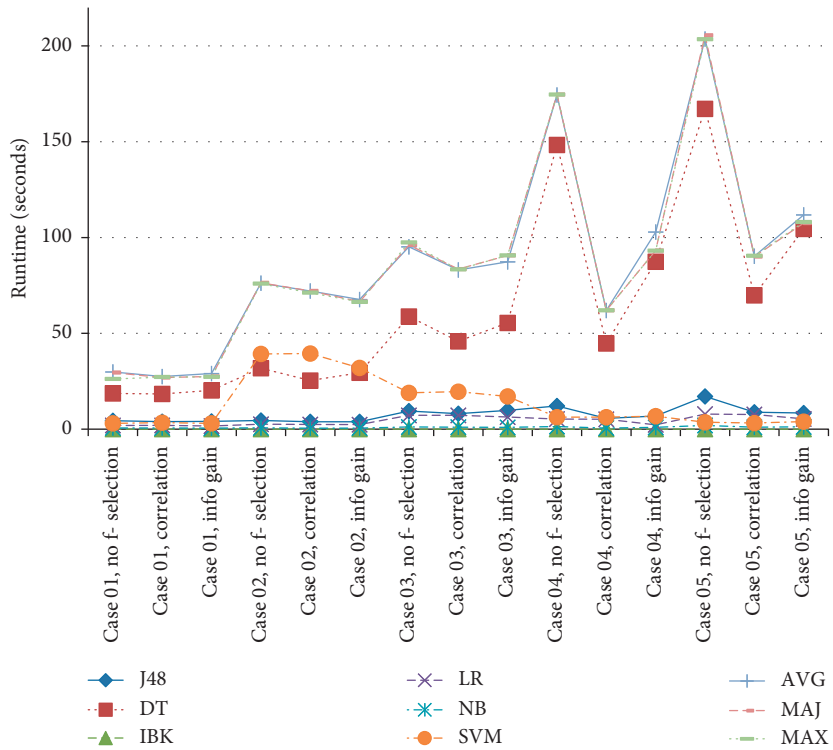


Figure 7: Runtime comparison of 9 classifiers on 5 cases before and after feature selection.

Table 14: Accuracy and efficiency of proposed adaptive model.

Case ID	Adaptive (before)	Adaptive (after)	Accuracy (before)	Accuracy (after)	Runtime (before)	Runtime (after)
01	J48	J48	95.93	*95.96*	4.43	4.05
02	J48	J48	94.72	94.66	4.54	3.86
03	AVG	AVG, MAJ	96.43	*96.45*	95.18	87.31, & 90.63
04	AVG, MAJ	AVG	90.64	*90.77*	174.4, & 174.6	102.86
05	MAJ	AVG	95.69	*95.80*	205.50	111.80

True positive (TP) is the amount of correct positive prediction; false positive (FP) is the incorrect positive prediction; true negative (TN) is the amount of correct negative prediction; and false negative (FN) is the amount of incorrect negative prediction. These four outcomes form the confusion matrix as shown in Figure 8.

The evaluation of effectiveness on our proposed model by means of accuracy, precision, and sensitivity is described in Table 15. According to the results shown in Table 15, our adaptive model achieves a good detection accuracy for the phishing features. Meanwhile, the performance of all the classifiers gets an acceptable precision and sensitivity ratio. According to the previous experiments, our adaptive phishing detection model using case-based reasoning can perform well on the diversely distributed features.

		Predicted	
		Positive	Negative
Actual	Positive	TP	FN
	Negative	FP	TN

Figure 8: Confusion matrix.

Table 15: Detection results achieved by the proposed model.

Case	Classifier	Accuracy (%)	Precision (%)	Sensitivity (%)
01	J48	95.96	83	79
02	J48	94.66	87	86
03	AVG	96.45	92	75
04	AVG	90.77	84	62
05	AVG	95.80	90	74

5. Conclusions

An adaptive mobile phishing detection model based on a variation of input feature patterns using a case-based reasoning (CBR) technique is proposed in this work. An experimental analysis is conducted to demonstrate the design decision of our model and to verify the performance of our proposed model in handling the concept drift of mobile phishing attacks. The proposed model is evaluated with a large feature set that contains 1,065 features from 10 feature

groups which are frequently collected from Android apps. Moreover, 5 cases of randomly combined patterns of features are created in order to provide a diversity of unknown patterns to mimic new real-world mobile apps. Six classification algorithms are chosen from different categories for the coverage usage of all classification nature on the diversion of feature sets. Three ensembles of six base classifiers are used, each of which uses different final answer-finding methods including average, majority voting, and maximum. In total, there are 9 classifiers. Due to the involvement of efficient features in the dataset and the uses of multiple classifiers, the efficiency degradation happened. To overcome this hurdle, 2 feature selection techniques are applied on the dataset in order to reduce the size of the features which is the size of the input to the classifiers. The two feature selection techniques used are information gain attribute evaluation method and Pearson's correlation coefficient attribute evaluation method. By addressing the optimal selection of the suitable classifier to the incoming features using a case-based reasoning approach, the proposed mobile phishing detection model could provide an accuracy improvement with an acceptable runtime increment.

Data Availability

The dataset of the features used in this research is available from the authors upon request.

Conflicts of Interest

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

Acknowledgments

This research was supported by the Higher Education Research Promotion and the Thailand's Education Hub for Southern Region of ASEAN Countries Project Office of the Higher Education Commission.

References

[1] W. Paul, H. A. Manolian, and S. Lapper, "Thinking digital in industry 4.0," Deloitte Insights, September 2018, https://www2.deloitte.com/insights/us/en/focus/industry-4-0/digital-leaders-in-manufacturing-fourth-industrial-revolution.html.

[2] "Spam and phishing in Q2 2018," Securelist-Kaspersky Lab's Cyberthreat Research and Reports, 2018.

[3] Proofpoint Security Awareness, "2019 state of the phish report," March 2019, https://www.wombatsecurity.com/state-of-the-phish.

[4] L. Wu, X. Du, and J. Wu, "Effective defense schemes for phishing attacks on mobile computing platforms," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 8, pp. 6678–6691, 2016.

[5] M. Moghimi and A. Y. Varjani, "New rule-based phishing detection method," *Expert Systems with Applications*, vol. 53, pp. 231–242, Jul. 2016.

[6] Baunfire.com and SparkCMS, "APWG phishing attack trends report-4Q 2018," Anti-Phishing Working Group, March 2019, https://www.antiphishing.org/resources/apwg-reports/.

[7] R. Basnet, S. Mukkamala, and A. H. Sung, "Detection of phishing attacks: a machine learning approach," in *Soft Computing Applications in Industry*, B. Prasad, Ed., pp. 373–383, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[8] A. K. Jain and B. B. Gupta, "Comparative analysis of features based machine learning approaches for phishing detection," in *Proceedings of the 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 2125–2130, New Delhi, India, March 2016.

[9] F. Toolan and J. Carthy, "Phishing detection using classifier ensembles," in *Proceedings of the 2009 eCrime Researchers Summit*, pp. 1–9, Tacoma, WA, USA, October 2009.

[10] H. S. Hota, A. K. Shrivas, and R. Hota, "An ensemble model for detecting phishing attack with proposed remove-replace feature selection technique," *Procedia Computer Science*, vol. 132, pp. 900–907, 2018.

[11] *A Comparative Study of Phishing Websites Classification Based on Classifier Ensembles*, ResearchGate, Berlin, Germany, 2019, https://www.researchgate.net/publication/325483941_A_Comparative_Study_of_Phishing_Websites_Classification_Based_on_Classifier_Ensembles.

[12] W. Wang, Y. Li, X. Wang, J. Liu, and X. Zhang, "Detecting Android malicious apps and categorizing benign apps with ensemble of classifiers," *Future Generation Computer Systems*, vol. 78, pp. 987–994, 2018.

[13] A. Aleroud and L. Zhou, "Phishing environments, techniques, and countermeasures: a survey," *Computers and Security*, vol. 68, pp. 160–196, 2017.

[14] H. Shahriar, T. Klintic, and V. Clincy, "Mobile phishing attacks and mitigation techniques," *Journal of Information Security*, vol. 6, no. 3, pp. 206–212, 2015.

[15] T. M. Mahmoud and A. M. Mahfouz, "SMS spam filtering technique based on artificial immune system," *International Journal of Computer Science Issues*, vol. 9, no. 1, pp. 589–597, 2012.

[16] J. W. Yoon, H. Kim, and J. H. Huh, "Hybrid spam filtering for mobile communication," *Computers and Security*, vol. 29, no. 4, pp. 446–459, 2010.

[17] C. H. Hsu, P. Wang, and S. Pu, "Identify fixed-path phishing attack by STC," in *Proceedings of the 8th Annual Collaboration, Electronic Messaging, Anti-Abuse and Spam Conference*, pp. 172–175, Perth, Australia, September 2011.

[18] E. Medvet, E. Kirda, and C. Kruegel, "Visual-similarity-based phishing detection," in *Proceedings of the 4th International Conference on Security and Privacy in Communication Networks*, Istanbul, Turkey, September 2008.

[19] A. P. Felt and D. Wagner, *Phishing on Mobile Devices*, University of California, Berkeley, CA, USA, 2011.

[20] A. Bianchi, J. Corbetta, L. Invernizzi, Y. Fratantonio, C. Kruegel, and G. Vigna, "What the app is that? Deception and countermeasures in the android user interface," in *Proceeding of the 2015 IEEE Symposium on Security and Privacy*, pp. 931–948, San Jose, CA, USA, May 2015.

[21] C. Marforio, R. J. Masti, C. Soriente, K. Kostiainen, and S. Capkun, "Personalized security indicators to detect application phishing attacks in mobile platforms," February 2015, http://arxiv.org/abs/150206824.

[22] D. Liu, E. Cuervo, V. Pistol, R. Scudellari, and L. P. Cox, "ScreenPass: secure password entry on touchscreen devices," in *Proceeding of the 11th Annual International Conference on*

Mobile Systems, Applications, and Services, pp. 291–304, Taipei, Taiwan, June 2013.

[23] D. Liu and L. P. Cox, "VeriUI: Attested Login for Mobile Devices," in *Proceedings of the 15th Workshop on Mobile Computing Systems and Applications*, Santa Barbara, CA, USA, February 2014.

[24] L. Wu, X. Du, and J. Wu, "MobiFish: A lightweight anti-phishing scheme for mobile phones," in *Proceedings of the 2014 23rd International Conference on Computer Communication and Networks (ICCCN)*, pp. 1–8, Shanghai, China, August 2014.

[25] V. Mavroeidis and M. Nicho, "Quick response code secure: a cryptographically secure anti-phishing tool for QR code attacks," in *Computer Network Security*, pp. 313–324, 2017.

[26] "Phishing detective-apps on Google Play," March 2018, https://play.google.com/store/apps/details?id=com.rsoftr.android.phishingdetectiveads.

[27] G. Bottazzi, E. Casalicchio, D. Cingolani, F. Marturana, and M. Piu, "MP-Shield: A framework for phishing detection in mobile devices," in *Proceedings of the 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, pp. 1977–1983, Liverpool, UK, October 2015.

[28] M. M. Richter and R. O. Weber, *Case-Based Reasoning*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[29] S. Craw, N. Wiratunga, and R. C. Rowe, "Learning adaptation knowledge to improve case-based reasoning," *Artificial Intelligence*, vol. 170, no. 16-17, pp. 1175–1192, Nov. 2006.

[30] S. Begum, M. U. Ahmed, P. Funk, N. Xiong, and M. Folke, "Case-based reasoning systems in the health sciences: a survey of recent Trends and developments," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 41, no. 4, pp. 421–434, Jul. 2011.

[31] S. Arzt, "FlowDroid: precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps," in *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 259–269, New York, NY, USA, June 2014.

[32] L. Li, A. Bartel, T. F. Bissyande et al., "IccTA: detecting inter-component privacy leaks in android apps," in *Proceedings of the 37th International Conference on Software Engineering*, vol. 1, pp. 280–291, Piscataway, NJ, USA, May 2015.

[33] Obfuscation-resilient, efficient, and accurate detection and family identification of android malware—semantic scholar," March 2018, https://paper/Obfuscation-Resilient%2C-Efficient%2C-and-Accurate-and-Garcia-Hammad/959093db69abc3b0fb4f7acc696a7f6ef39d0e23.

[34] W. Enck, "TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones," *Transactions on Computer Systems*, vol. 32, no. 2, 2014.

[35] M. I. Gordon, D. Kim, J. Perkins, L. Gilham, N. Nguyen, and M. Rinard, "Information-flow analysis of android applications in DroidSafe," in *Proceedings of the Network and Distributed System Security Symposium*, San Diego, CA, USA, February 2015.

[36] D. Arp, M. Spreitzenbarth, H. Gascon, and K. Rieck, "Drebin: effective and explainable detection of android malware in your pocket," in *Proceedings of the 2014 Network and Distributed System Security Symposium*, San Diego, CA, USA, February 2014.

[37] N. Peiravian and X. Zhu, "Machine learning for android malware detection using permission and API calls," in *Proceedings of the 2013 IEEE 25th International Conference on Tools with Artificial Intelligence*, pp. 300–305, Herndon, VA, USA, November 2013.

[38] V. Avdiienko, K. Kuznetsov, A. Gorla et al., "Mining apps for abnormal usage of sensitive data," in *Proceedings of the 37th International Conference on Software Engineering*, vol. 1, pp. 426–436, Florence, Italy, May 2015.

[39] H. V. Nath and B. M. Mehtre, "Static malware analysis using machine learning methods," in *Recent Trends in Computer Networks and Distributed Systems Security*, pp. 440–450, 2014.

[40] N. Abura'ed, H. Otrok, R. Mizouni, and J. Bentahar, "Mobile phishing attack for Android platform," in *Proceedings of the 2014 10th International Conference on Innovations in Information Technology (IIT)*, pp. 18–23, Abu Dhabi, UAE, November 2014.

[41] VirusShare.com, https://virusshare.com.

[42] K. Allix, T. F. Bissyandé, J. Klein, and Y. Le Traon, "Androzoo: collecting millions of android apps for the research community," in *Proceedings of the 13th International Conference on Mining Software Repositories*, pp. 468–471, Austin, TX, USA, May 2016.

[43] J. Yu, Q. Huang, and C. Yian, "DroidScreening: a practical framework for real-world Android malware analysis," *Security and Communication Networks*, vol. 9, no. 11, pp. 1435–1449.

[44] Joshuaga/Revealdroid—Bitbucket, https://bitbucket.org/joshuaga/revealdroid/src/master/.

[45] S. Kyaw Zaw and S. Vasupongayya, "Revealing the important features of mobile phishing," in *Proceedings of the 13th International Conference on Knowledge, Information and Creativity Support Systems (KICSS 2018)*, pp. 222–226, Pattaya, Thailand, November 2018.

[46] M. A. Hall and L. A. Smith, "Feature subset selection: a correlation based filter approach," *Progress in Connectionist-based Information Systems*, vol. 2, pp. 855–858, 1997.

Hindawi
Journal of Computer Networks and Communications
Volume 2019, Article ID 4708201, 13 pages
https://doi.org/10.1155/2019/4708201

Research Article

Multistage System-Based Machine Learning Techniques for Intrusion Detection in WiFi Network

Vu Viet Thang[1] **and F. F. Pashchenko**[2,3]

[1]*Moscow Institute of Physics and Technology (State University), Moscow, Russia*
[2]*The Department of Information and Communication Technologies, MIPT (State University), Moscow, Russia*
[3]*Trapeznikov Institute of Control Sciences, Russian Academy of Sciences, Moscow, Russia*

Correspondence should be addressed to Vu Viet Thang; vv.tkhang@phystech.edu

Received 6 December 2018; Accepted 9 April 2019; Published 28 April 2019

Guest Editor: Arash H. Lashkari

The aim of machine learning is to develop algorithms that can learn from data and solve specific problems in some context as human do. This paper presents some machine learning models applied to the intrusion detection system in WiFi network. Firstly, we present an incremental semisupervised clustering based on a graph. Incremental clustering or one-pass clustering is very useful when we work with data stream or dynamic data. In fact, for traditional clustering such as K-means, Fuzzy C-Means, DBSCAN, etc., many versions of incremental clustering have been developed. However, to the best of our knowledge, there is no incremental semisupervised clustering in the literature. Secondly, by combining a K-means algorithm and a measure of local density score, we propose a fast outlier detection algorithm, named FLDS. The complexity of FLDS is $O(n^{1.5})$ while the results obtained are comparable with the algorithm LOF. Thirdly, we introduce a multistage system-based machine learning techniques for mining the intrusion detection data applied for the 802.11 WiFi network. Finally, experiments conducted on some data sets extracted from the 802.11 networks and UCI data sets show the effectiveness of our new proposed methods.

1. Introduction

Machine learning is a central problem in artificial intelligence. The purpose of machine learning is concerned with the development of algorithms and techniques that allow computers to *learn*. There are some principal kinds of machine learning such as supervised learning, unsupervised learning, and semisupervised learning. The application of machine learning techniques is very varied, for example, fault detection in bank data, transaction data, and intrusion detection system in networking, bioinformatics, natural language processing, image analysis, etc. [1]. Additionally, machine learning is very useful in cases in which human expertise does exist (robot in the Mars, in the sea, etc.), solution change in time (networking, surveillance), or solution needs to be adapted to particular cases. This paper focuses on developing machine learning techniques for intrusion detection systems in WiFi network.

Intrusion detection system (IDS) is one of the most emerging tasks in the network connectivity. Each year, there are lots of network attacks in the world; consequently, the cost for solving these problems is very big, and was reported to be about 500 billion USD in 2017. This problem is a challenge not only for government/organizations but also for individuals in daily lives. To protect the computer network system, in general, some methods can be used such as firewalls, data encryption, or user authentication. The firmware is one technique to protect the system, but nowadays, the external mechanisms have emerged and quickly become popular. One important method for data mining in intrusion detection problem proposed in the literature is to use machine learning techniques [2–8]. The IDS has monitored directly the network transactions where each transaction is either normal or malicious. The aim of IDS is to detect and alert network administrators when it detects a transaction that

is an attack. In some case, the IDS can even immediately block the connection.

Generally, data mining task in IDS must detect two kinds of attack including known attacks and outlier (anomaly) attacks. For the known attacks, we can use a (semi-)supervised learning method such as neural network, support vector machine, random forest, decision tree, and naïve Bayes, to mention a few, to construct a classifier from data training (labeled normal/attacks connection) [4–7, 9]. The classifier trained is used for detecting new connections, and the supervised learning model is illustrated in Figure 1. With the outlier attacks in which we do not know its labels, the trained classifier cannot detect them. In this case, we have to use another kind of machine learning called unsupervised outliers detection such as LOF [10], ODIN [11], and so on. The outliers detection process can be realized offline for some periods of time defined by users/experts. The general schema for outlier detection is presented in Figure 2, and this is the unsupervised learning model. The aim of this schema is to detect outliers in a period of time. For example of IDS systems, the users can set a period of time from u to v for capturing the data, then the data will be transformed by the preprocessing step, and finally, we can use an outlier detection method to detect attacks from the observed data.

The contributions of our paper are as follows:

(i) We propose an incremental semisupervised graph-based clustering. To the best of our knowledge, this is the first incremental semisupervised clustering algorithm. The preliminary work is presented in [12].

(ii) We introduce a fast outliers detection method based on local density score and K-means clustering algorithm. The preliminary work is introduced in [13].

(iii) We propose a multistage system-based machine learning techniques which can boost the accuracy of the intrusion detection process for the 802.11 WiFi data set.

(iv) The experiments carefully conducted on data set extracted from Aegean WiFi Intrusion Dataset (AWID) show the effectiveness of our proposed algorithms [14]. The AWID is a publicly available collection of sets of data which contain real traces of both the normal and intrusive 802.11 traffic. Up to date, AWID is one of the standard data sets to evaluate the capacity of IDS systems.

This paper is organized as follows. Section 2 presents the related work. Section 3 introduces the new incremental semisupervised clustering method and a new fast outlier detection algorithm. Section 4 presents experiments for the proposed algorithms and proposes a hybrid framework applied for the AWID data set. Finally, Section 5 concludes the paper and presents some direction for further research studies.

2. Incremental Clustering and Outlier Detection

2.1. Incremental Clustering. Clustering is the task of partitioning a data set into k clusters in which the points in the same cluster are similar and the points in different clusters are dissimilar. The context of incremental clustering is as follows: given some current clusters, the incremental clustering is one-pass clustering kind which aims to identify cluster label for incremental data points. Incremental clustering is very useful for data stream or dynamic data (data warehouse). In general, the incremental clustering is combined with two processes of insertion and deletion. Given a set of clusters, the insertion step aims to identify the labels of a new data point based on the current clusters. In some cases, some new clusters will be created or the new data points will be integrated with the current clusters. With the deletion process, if we want to remove one or some data points, we need to reform the clusters because some clusters may be affected by these operations. For each kind of clustering, there are some incremental clustering algorithms proposed in the literature such as Incremental K-means [15], IncrementalDBSCAN [16], or Incremental graph clustering [17]. The key idea of these algorithms is that we need to identify the situation for each kind of algorithm for the insertion step and deletion step. The incremental clustering addresses the problem of identifying the label for a new data object or updating clusters when we remove points in the current clusters. This problem is very meaningful when we tackle with the big data in which the data set is too big to fit into the available memory. For each kind of clustering, there are some versions of incremental clustering proposed in the literature.

In [16], the Incremental density-based clustering (IncrementalDBSCAN) is introduced. Based on the notion of density-based clustering, the IncrementalDBSCAN can efficiently add and delete points for the current clusters. The adding process of a new point has some cases; for example, the new point can be noise, the new point will be added in a cluster, and the new point can merge some clusters. For the deletion process, the point can be a noise point and the point can split to some clusters or not affect the current clusters. Some cases of the insertion process and deletion process of IncrementalDBSCAN are shown in Figure 3.

In [15], a single-pass incremental clustering for large data set based on K-means is introduced (named GenIC). GenIC updates each center with each new data point and merges clusters only at the end of a generation (i.e., window of data). By a generalized incremental algorithm, GenIC algorithm can move a center in the list of centers using a weighted sum of the existing center and the new point presented. The idea of GenIC is to divide the stream of data into chunks or windows as is common with streaming algorithms. We view each chunk of n data points as a generation and think of the "fitness" of a center as being measured by the number of points assigned to it. In general, the fittest centers survive to the next generation, but occasionally new centers are selected and old centers are killed off. The GenIC is compared with K-means and shown the effectiveness in running time and less affected by the choice of initial centers than K-means. In [18], a version of Incremental K-means clustering is also proposed. In the

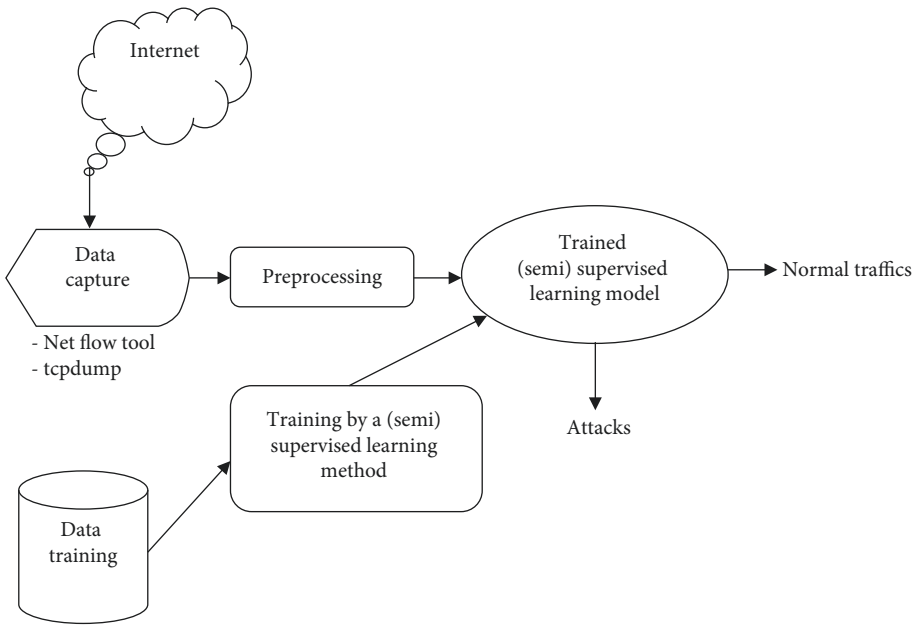


Figure 1: A general model for misuse detection in IDS.

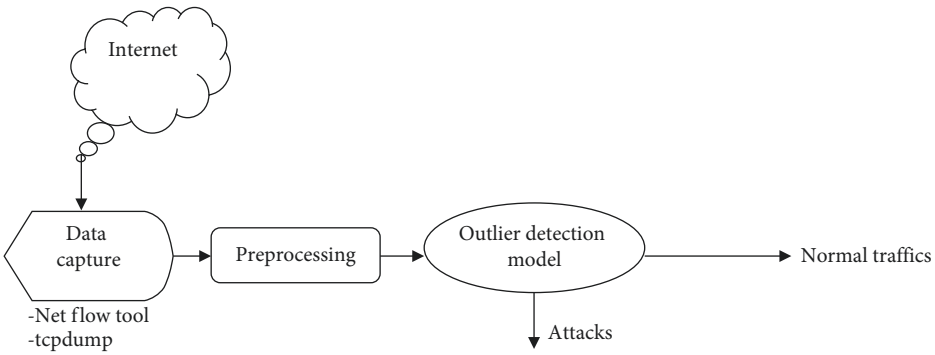


Figure 2: A general model for outlier detection in IDS.

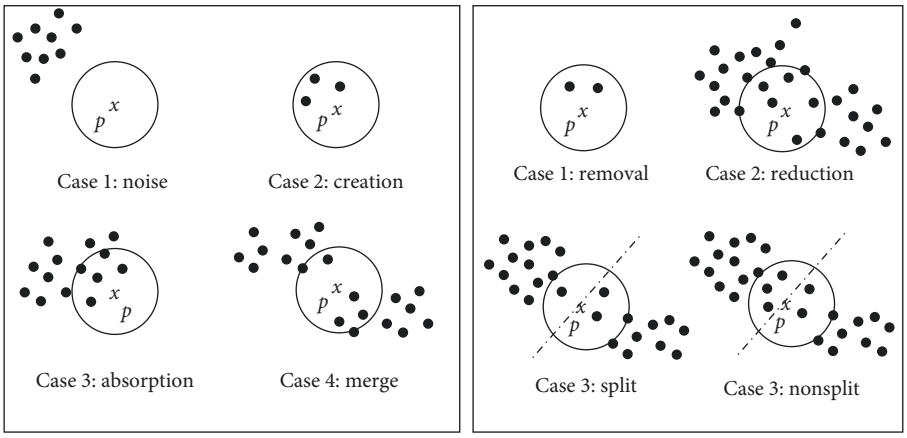


Figure 3: Insertion cases (a) and deletion cases (b) of IncrementalDBSCAN [16].

algorithm, clusters are built incrementally by adding one cluster center at a time. In [19], a novel two-phase static single-pass algorithm as well as a dynamic two-phase single-pass algorithm based on Fuzzy C-means have been presented and are showing high utility. The idea behind the multistage methods reported in the paper is that an estimate of the partition matrix and the location of the cluster centers can be obtained by clustering a sample of the data. A small sample is expected to produce a fast yet less reliable estimation of the cluster centers. This leads to a

multistage approach, which involves several stages of sampling (with replacement) of the data and estimating the membership matrix for the next stage. The experiments conducted show the effectiveness of the proposed method. In [17], Chandrasekhar et al. propose an incremental local density clustering scheme for finding dense subgraphs in streaming data, i.e., when data arrive incrementally (ILDC). The incremental clustering scheme captures redundancy in the streaming data source, by finding dense subgraphs, which correspond to salient objects and scenes. The ILDC process performs greedy operations like cluster expansion, cluster addition and cluster merging based on the similarity between clusters defined. The ILDC shows the effectiveness when using in image-retrieval applications. In [20], an incremental semisupervised ensemble clustering algorithm has successfully presented, named ISSCE. ISSCE uses constraints to update incremental members. The authors develop an incremental ensemble member selection process based on a global objective function and a local objective function to remove the redundant ensemble members. The experiment results show the improvement of ISSCE over traditional semisupervised clustering ensemble approaches or conventional cluster ensemble methods on six real-world datasets from UCI machine learning repository and 12 real-world data sets of cancer gene expression profiles. In the context of classification, we need to find the label for a new data object by using a classifier trained by data training. The problem of identifying the label for a new object in incremental clustering can be seen similar to classification context.

2.2. Outlier Detection Problem. Outlier (anomaly) detection is one of the important problems of machine learning and data mining. As mentioned in [21], outliers detection is the problem of finding patterns in data that do not conform to expected behavior. The applications of outlier detection can be found in many applications such as intrusion detection, credit fraud detection, video surveillance, weather prediction, discovery of criminal activities of electronic commerce, etc. [9, 21]. There are some kinds of outliers including point outliers, contextual outliers, and collective outliers. In this paper, we focus on point outliers detection that can be applied in a variety of applications. For a data set consisting of points, a point will be called outlier if it is different from a large number of the rest of the points. To detect outliers, there are some principal methods in the literature such as classification methods, nearest neighbor methods, clustering methods, statistical methods, distance-based methods, etc.

For the classification-based outliers detection, we have two categories: multiclass and one-class anomalies detection methods. In multiclass classification techniques, we assume that the training data contain labeled points of all normal classes. The learner using a supervised learning model trains a model using the labeled data. The classifier can distinguish between each normal class and the rest of the class. A test point will be called outlier if it does not belong to any normal class. In one-class outliers detection methods, we assume that the number of normal class is only one. The classifier learns a model that can detect the boundary of the normal class. If a test point does not fall in the boundary, it will be called outliers. Although many techniques have been done, however, the main disadvantage of these methods based on the availability of accurate labels for normal classes which is not easy to apply for real applications.

For the nearest neighbor-based outlier detection methods, we use the assumption as follows: normal points belong to the dense regions, while outliers belong to the sparse regions. The most famous method of this kind is the LOF algorithm. The idea of LOF is based on the local density evaluation score for points. Each point will be assigned a score which is the ratio of the average local density of the k-nearest neighbors of the point and the local density of the data point itself. Many variants of LOF can be cited here such as COF [22], ODIN [11], LOCI [23], etc. The main drawback of the method is the $O(n^2)$ complexity required.

For the clustering-based outliers detection techniques, the idea here is using clustering methods to group data into clusters. The points do not belong to any clusters called outliers. Some clustering methods can be detected outliers such as DBSCAN [24], SNN [25], etc. In fact, the purpose of clustering is finding clusters, so the outliers are just the product of the clustering process and hence are not carefully optimized. One more reason that can be made here is the complexity of clustering techniques required $O(n^2)$.

In the statistical outliers detection methods, these methods are based on the assumption as follows: normal data points occur in high-probability regions of a stochastic model, while anomalies occur in the low-probability regions of the stochastic model. Some methods have been done for the kind of outliers detections. In general, statistical methods fit a statistical model (Gaussian distribution, the mixture of parametric statistical distribution, etc.) to the given data and then apply a statistical inference test to determine if an unseen instance belongs to this model or not. The key limitation of these methods is the assumption about the distribution of data points. This assumption is not true, especially when the dimension of data is high [21].

In the distance-based outliers detection methods, a point is considered as outlier if it does not have enough pct% points in the data set that distance from this point is smaller than the threshold value $d_{\min}$ [26].

3. Proposed Method

3.1. Semisupervised Graph-Based Clustering. In recent years, semisupervised clustering is an important research topic that is illustrated by a number of studies introduced [27]. The purpose of semisupervised clustering is to integrate side information for improving the clustering performances. Generally, there are two kinds of side information including constraints and seeds. Given a data set X, constraints involve must-link and cannot-link in which the must-link constraint

(ML) between two observations $x \in X$ and $y \in X$ means that x and y should be in the same cluster, and the cannot-link constraint (CL) means that x and y should not be in the same cluster. With seeds, a small set of labeled data (called seeds) $S \in X$ will be provided for semisupervised clustering algorithms. In fact, this side information is available or can be collected from users [28–31]. We can cite here the work of semisupervised clustering for K-means [32], hierarchical clustering [33], graph-based clustering [34, 35], spectral clustering [36, 37], density-based clustering [38], etc. While many semisupervised clustering algorithms are introduced, to the best of our knowledge, there are no incremental semisupervised clustering algorithms in the literature.

Our new incremental clustering introduced in the next section is based on the work of semisupervised graph-based clustering using seeds (SSGC). We choose the SSGC algorithm because SSGC algorithm has several advantages such as SSGC use only one parameter and SSGC can detect clusters in varied density regions of data [35]. SSGC includes two steps as the following description (see Algorithm 1):

Step 1. Given a k-nearest neighbor graph presenting a data set X, this step uses a loop in which at each step, all edges which have the weight less than threshold θ will be removed. The value of θ is initialized by 0 at the first step and incremented by 1 after each step. This loop will stop when each connected component has at most one kind of seeds. The main clusters are identified by propagating label in each connected component that contains seeds.

Step 2. The remaining points (graph nodes) that do not belong to any main clusters will be divided into two kinds: points that have edges which relate to one or more clusters and other points which are isolated points. In the first case, points will be assigned to the cluster with the largest related weight. For the isolated points, we can either remove them as outliers or label them.

We note that, in SSGC, the weight $\omega(x_i, x_j)$ of the edge (the similarity) between two points x_i and x_j in the k-nearest neighbor graph is equal to the number of points that the two points share, as the following equation:

$$\omega(x_i, x_j) = \left|\mathrm{NN}(x_i) \cap \mathrm{NN}(x_j)\right|, \tag{1}$$

where $\mathrm{NN}(\cdot)$ is the set of k-nearest neighbors of the specified point.

SSGC is efficient when compared with the semi-supervised density-based clustering in detecting clusters for batch data; however, it is not adapted for data stream or data warehousing environment where many updates (insertion/deletion) occur.

3.2. Incremental Graph-Based Clustering Using Seeds. In this section, we propose IncrementalSSGC, based on the SSGC algorithm. In the IncrementalSSGC, the seeds will be used to train a k-nearest neighbor graph to construct connected components and identify the value of θ as in SSGC algorithm. Like other incremental clustering algorithms, two procedures must be developed, including insertion and deletion.

Algorithm 2 shows the insertion step of IncrementalSSGC for a new data point x_{new}. At first, the list of edges between x_{new} and the current clusters is created, and all edges with weight smaller than θ will be removed. If the list is empty, it is illustrated that x_{new} is an outlier with the current situation, and hence, x_{new} will be added in a temporary list Lo. In the case of existing edges between x_{new} and some connected components, we need to remove some edges until x_{new} connects to components with one kind of label. Finally, the label of x_{new} will be identified by the label of its connected components. In Step 10, x_{new} and its related edges will be added to L; some edges between x_{t} and x_{l} will also be recalculated if x_{new} appears in the nearest neighbors list of x_t or x_l. In Step 12, after some insertion steps, we can examine the points in Lo.

Algorithm 3 presents the detailed steps of the deletion process. When we want to remove a point x_{del} from the current clusters, we simply remove x_{del} and all edges related with x_{del} in the graph. Step 2 of the algorithm shows the updating process. In this step, we need to update all edges affected by x_{del}. It means that all edges between x_i and x_j must be updated if x_{del} appears in the commune list of the nearest neighbors. Finally, Step 3 is simply to remove all edges that have weight less than θ.

3.2.1. The Complexity Analysis. Now, we will analyse the complexity of IncrementalSSGC. Given a data set with n object, we recall that the complexity of SSGC is $O(k \times n^2)$, in which k is the number of nearest neighbors. Assuming that we have the current clusters including n objects, we will analyse the complexity of the insertion and deletion process of IncrementalSSGC at step $(n+1)$ as follows.

For the insertion process which aims to identify the cluster label for a new data point x_{new}, in Step 1, to create the list of edges between x_{new} and the current clusters, the complexity is $O(n \times k)$. In Steps 2, 6, and 7, the complexity is just $O(k)$. In Step 10, some edges between x_t and x_l will also be recalculated if x_{new} appears in the nearest neighbors list of x_t or x_l; in fact, the number of such edges is also small. So for the insertion of a new point, the complexity is $O(n \times k)$.

For the deletion process, the complexity of Step 1 is $O(k)$. In Steps 2 and 3, the number of edges updated is the number of edges that received x_{del} as commune points, and the value of commune points depends on the data set. Let q be the average value of v deletion processes; in fact, q is determined by perfoming experiments. So, the complexity of a deletion process is $O(q \times n \times k)$.

In summary, with the analysis of the insertion and deletion process above, we can see that it is very useful for data set that we usually need to update. In the next section, we also present the running time of both SSGC and IncrementalSSGC for some data sets extracted from intrusion detection problem.

Input: $\mathcal{X}$, number of neighbors k, a set of seeds $\mathcal{S}$
Output: A set of detected clusters/outliers
PROCESS:
(1) Constructing the k-NN graph of $\mathcal{X}$
(2) $\theta = 0$
(3) **repeat**
(4) Constructing the connected components using the threshold θ
(5) $\theta = \theta + 1$
(6) **until** the *cut condition* is satisfied
(7) Propagating the labels to form the *principal clusters*
(8) Constructing the final clusters

Algorithm 1: The algorithm SSGC [35].

Input: a new data object x_{new}; a set of current clusters C, list containing edges for each point of current clusters L, θ (threshold), and number of nearest neighbors (NN) k.
Output: label for x_{new}
Process:
(1) Create the k-nearest neighbors list of edges (LE) between x_{new} and all current clusters
(2) Delete all $(u, v) \in \text{LE}$: $\text{weight}(u, v) < \theta$
(3) **if** $(\text{LE} = \varnothing)$ **then**
(4) x_{new} is added in a temporary list Lo
(5) **else**
(6) **If** x_{new} related to two or more components with different label then
(7) Delete edges in LE with ascending order of weight until x_{new} connecting with components with at most one kind of label
(8) **end if**
(9) Get label for x_{new} and its connected points (if any) by propagating
(10) Update list L: adding edges relating to x_{new} to L; some edges between x_t and x_l will also be recalculated if x_{new} appears in the nearest neighbors list of x_t or x_l.
(11) **end if**
(12) Examine points in Lo

Algorithm 2: IncrementalSSGC: insertion process.

Input: an object x_{del} in a component will be deleted; a set of current clusters C, list of edges for each point of current clusters L, θ (threshold)
Output: the updated C, the updated L,
Process:
(1) Delete x_{del} and all edges related to x_{del} in L
(2) Update all weights $(k, l) \in C$: $x_{\text{del}} \in \text{NN}(k) \cap \text{NN}(l)$
(3) Delete all updated (at Step 2) $(k, l) \in L$: $\text{weight}(t, l) < \theta$

Algorithm 3: IncrementalSSGC: deletion process.

3.3. A Fast Outlier Detection Method. Given a k-nearest neighbors graph (k-NNG), the local density score *LDS* of a vertex $u \in k$-NNG is defined as follows [39]:

$$\text{LDS}(u) = \frac{\sum_{q \in \text{NN}(u)} \omega(u, q)}{k}, \tag{2}$$

in which ω is calculated as in equation (1), and k is the number of nearest neighbors used. The LDS is used as an indicator of the density of the region of a vertex u. The LDS value is in the interval of $[0, k-1]$; the larger the LDS of u, the denser the region that u belongs to, and vice versa. So, we can apply the way of LDS's calculation to identify outliers. To detect outlier by this method, we have to use a parameter as the threshold: the point which has LDS value smaller than the threshold can be seen as an outlier and vice versa. Similar to LOF, the method has required $O(n^2)$ of complexity.

To reduce the running time of the method, we propose a Fast outlier detection method based on Local Density Score, called FLDS. The basic idea of the algorithm FLDS is to use divide-and-conquer strategy. Given a data set X to find outliers, first, the input data set will be split into k clusters using K-means algorithm. Next, k-nearest neighbor graphs will be used for each cluster and identify outlier on each local cluster. The outliers found in all clusters will be recalculated on the whole data set. The idea of divide-and-conquer strategies by using the K-means in the preprocessing step has been successfully applied in solving some problems such as fast spectral clustering problem [40] and fast minimum spanning tree problem [41] and in the efficient and effective shape-based clustering paper [42]. The FLDS algorithm is described in Algorithm 4.

The FLDS algorithm is an outlier's detection method based on K-means and local density score using graph. The complexity of FLDS is $O(n \times k) + O(k^2) + O(t \times n)$; in which the value of k may be used up to $n^{0.5}$ [41, 42]; $t \ll n$ is evaluated approximately equal to k; so the complexity of the FLDS is $O(n^{1.5})$.

4. Experiment Results

This section aims to evaluate the effectiveness of our proposed algorithms. We will show the results of the IncrementalSSGC, the results of FLDS, and the results when using our methods for a hybrid framework for intrusion detection problem. The IncrementalSSGC will be compared with the IncrementalDBSCAN, while the FLDS will be compared with the LOF.

The data sets used in the experiments are mostly extracted from the Aegean WiFi Intrusion Dataset (AWID) [14]. AWID is a publicly available collection of sets of data in an easily distributed format, which contain real traces of both the normal and intrusive 802.11 traffic. In the AWID, many kinds of attacks have been introduced, but they also fall into four main categories including flooding, injection, and impersonation. The AWID has 156 attributes, we use 35 attributes extracted by an artificial neural network, as presented in [8]. We also use some supplement data sets that come from UCI [43] and data sets with different size, shape, and density and contain noise points as well as special artifacts [44] in this experiment.

4.1. Experiment Setup

4.1.1. Data Sets for Incremental Clustering Algorithms. To show the effectiveness of the IncrementalSSGC, two aspects will be examined including the running time and accuracy. 5 UCI data sets and 3 data sets extracted from AWID will be used for testing IncrementalSSGC and IncrementalDBSCAN. The details of these data sets are presented in Table 1.

To evaluate clustering results, the Rand Index is used. Given a data set X with n points for clustering, P_1 is an array containing the true labels, P_2 is an array containing the results of a clustering algorithm, the Rand Index (RI) is calculated as follows:

$$\mathrm{RI} = \frac{a+b}{(n!/(2!(n-2)!))}, \tag{3}$$

in which a/b is the number of pairs that are in the same/different clusters in both partitions P_1 and P_2. The bigger the Rand Index, the better the result.

4.1.2. Data Sets for FLDS and LOF. We used 5 data sets extracted from AWDI and four 2D data sets including DS1 (10000 points), DS2 (8000 points), DS3 (8000 points), and DS4 (8000 points) [44] for FLDS and LOF. These 2D data sets have clusters of different size, shape, and orientation, as well as random noise points and special artifacts. The details of these AWID data sets are presented in Table 2.

To compare LOF and FLDS for AWID data sets, we use the ROC measure that has two factors including False Positive (False Alarm) Rate (FPR) and False Negative (Miss Detection) Rate (FNR). The detail of these factors is shown in the following equations:

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}, \tag{4}$$

$$\mathrm{FNR} = \frac{\mathrm{FN}}{\mathrm{TP} + \mathrm{FN}}, \tag{5}$$

in which True Positive (TP) is the number of attacks correctly classified as attack; True Negative (TN) is the number of normal correctly detected as normal; False Positive (FP) is the number of normal falsely classified as attacks, namely false alarm; and False Negative (FN) is the number of attacks falsely detected as normal.

To combine FPR and FNR values, we calculate the Half Total Error Rate (HTER) that is similar to the evaluation method used in [11], defined as follows:

$$\mathrm{HTER} = \frac{\mathrm{FPR} + \mathrm{FNR}}{2}. \tag{6}$$

4.2. Clustering Results. We note that there is no incremental semisupervised clustering algorithm in the literature. So we compare the performance obtained by our algorithm and the IncrementalDBSCAN algorithm. IncrementalDBSCAN can be seen as the state of the art among Incremental clustering proposed. The algorithm can detect clusters with different size and shape with noises. Because both SSGC and IncrementalSSGC produce the same results, we just show the results for IncrementalSSGC and IncrementalDBSCAN. The results are shown in Figure 4.

We can see from the figure that the IncrementalSSGC obtains better results compared with the IncrementalDBSCAN. It can be explained by the fact that the IncrementalDBSCAN cannot detect clusters with different densities as mentioned in the paper . . .*we assumed that the parameter values Eps and MinPts of DBSCAN do not change significantly when inserting and deleting objects*. . ..

Input: a data set X with n points, the number of nearest neighbors k, number of clusters nc, *theta*,
Output: outliers of X,
Process:
(1) Using K-means to split X into nc clusters
(2) Using LDS algorithm on each separate cluster to obtain local outliers (using the threshold *theta*)
(3) The local outliers obtained in Step 2 will be recalculated LDS's value across the data set

Algorithm 4: Algorithm FLDS.

Table 1: Main characteristics for clustering evaluation.

ID	Data	#Normal + impers.	#Attributes	#Clusters
1	Iris	150	4	3
3	Wine	178	13	3
2	*E coli*	336	8	8
4	Breast	569	30	2
5	Yeast	1484	8	10
6	AWID1	5000	35	2
7	AWID2	8000	35	2
8	AWID3	12000	35	2

Table 2: Main characteristics for FLDS and LOF.

ID	Data	#Objects	Categories
1	O-AWID1	3030	Impers., flooding, injections
3	O-AWID2	5030	Impers., normal, flooding
2	O-AWID3	7040	Flooding, normal, impers.
4	O-AWID4	10040	Normal, impers., injection, flooding
5	O-AWID5	15050	Normal, flooding, injection, and impers.

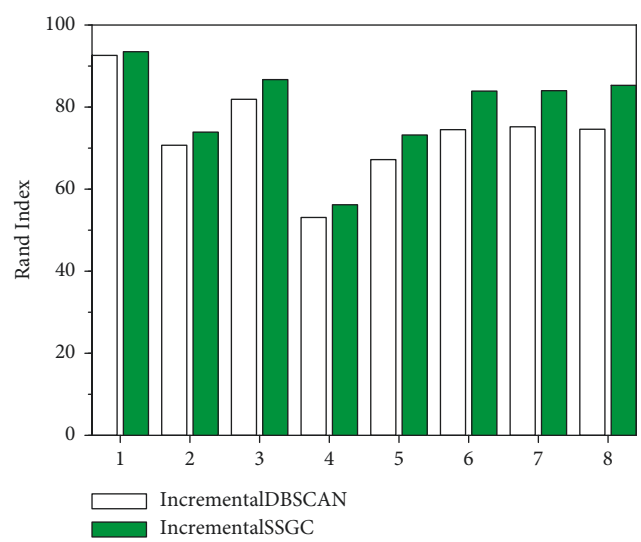


Figure 4: Clustering results obtained by IncrementalDBSCAN and IncrementalSSGC for 8 data set of Table 1, respectively.

This assumption means that the IncrementalDBSCAN cannot work well with the data set having different densities. In contrary to IncrementalDBSCAN, the algorithm IncrementalSSGC does not depend on the density of the data because the similarity measure used is based on shared nearest neighbors.

4.2.1. Running Time Comparison. Figure 5 presents the running time for IncrementalSSGC and IncrementalDBSCAN for three AWID data sets. We can see the running time of both algorithms is similar. It can be explained by the fact that both algorithms use k-nearest neighbor to find clusters for each step of incremental. We also present the running time of the SSGC algorithm for reference purpose. From this experiment, we can see advantages of the incremental clustering algorithms.

4.3. The Results of FLDS and LOF. Table 3 presents the results obtained by FLDS and LOF for 5 AWID data sets. We can see that the results of FLDS are comparable with the algorithm LOF. The parameters used for both methods are shown in Table 4. For some 2D data sets, Figure 6 presents the results obtained by FLDS and LOF. Intuitively, the outliers detected by both methods are mostly similar. We can explain the results by the fact that the strategy for evaluating a point is outlier or not based on local density score.

Figures 7 and 8 illustrate the running time comparison between FLDS and LOF. With 4 data sets mentioned above, it can be seen from the figure that the calculation time of FLDS is about 12 times faster than the running time of LOF. This is the significant improvement compared with LOF. It can be explained by the fact that the complexity of FLDS is just $O(n^{1.5})$ compared with $O(n^2)$ of LOF.

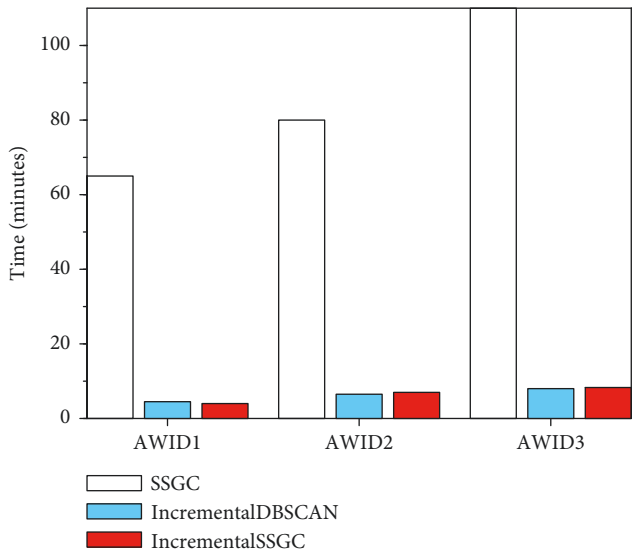


Figure 5: Running time comparison between IncrementalSSGC and IncrementalDBSCAN.

Table 3: The HTER measure of LOF and FLDS (the smaller, the better) for some extracted AWID data sets.

Methods	O-AWID1	O-AWID2	O-AWID3	O-AWID4	O-AWID5
FLDS	0.13	0.12	0.10	0.11	0.06
LOF	0.23	0.11	0.11	0.09	0.09

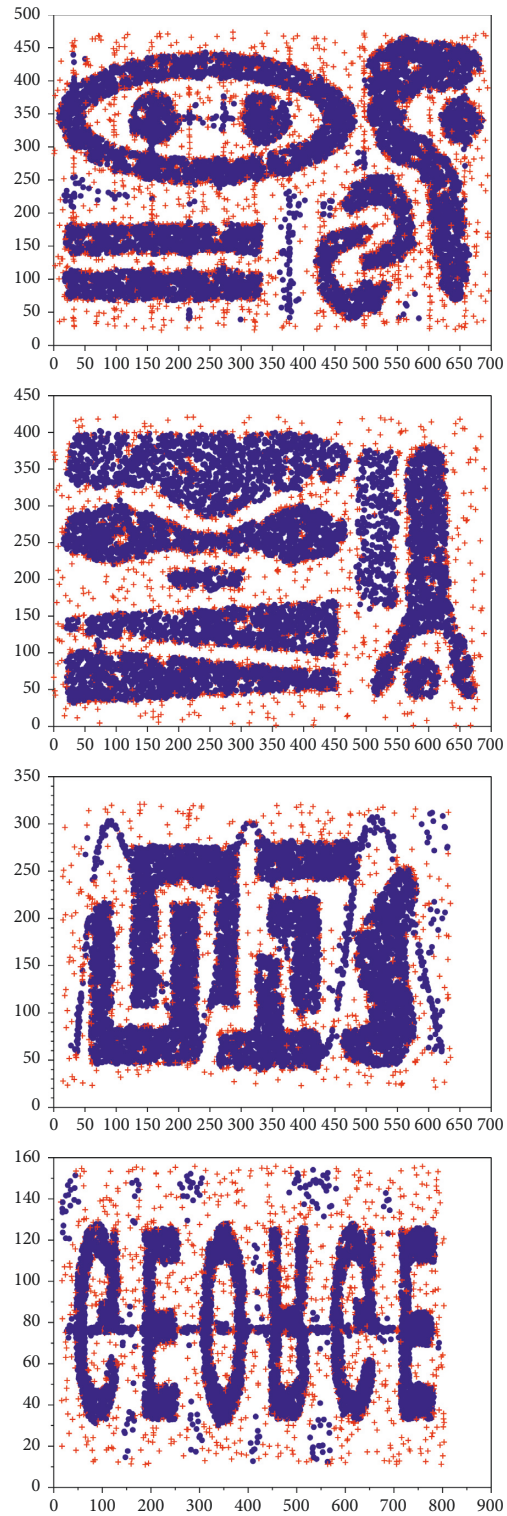
Table 4: The parameters used in data sets.

Methods	O-AWID1	O-AWID2	O-AWID3	O-AWID4	O-AWID5
FLDS (k, nc, θ)	(25, 30, 6)	(25, 30, 6)	(25, 30, 6)	(25, 45, 6)	(25, 45, 6)
LOF (MinPts, η)	(27, 1.2)	(27, 1.2)	(25, 1.2)	(25, 1.2)	(27, 1.2)

k: the number of nearest neighbors; nc: number of cluster used; θ: the threshold.

4.4. A Framework for Intrusion Detection in 802.11 Networks. In this section, we propose a multistage system-based machine learning techniques applied for the AWDI data set. The detail of our system is presented in Figure 9. Three components are used for intrusion detection task: a supervised learning model (J48, Bayes, random forest, support vector machine, neural network, etc.) trained by labeled data set, and this model can be seen as misuse detection component; an outlier detection method (LOF, FLDS, etc.) is optionally used to detect new attacks in some periods of time; additionally, for the AWID data sets as presented above, it is very difficult to detect impersonation attacks, so we use an Incremental clustering algorithm (IncrementalDBSCAN, IncrementalSSGC, etc.) for further finding this kind of attack.

In this experiment, we use J48 for the misuse detection process and IncrementalSSGC for the detecting impersonation attacks. In the outliers detection step, we propose to use FLDS or LOF, and the results have been presented in the subsection above. Because the outliers detection step can be realized offline for some periods of time, we just show the results obtained by combining J48 and IncrementalSSGC. The confusion matrix of



(a)

Figure 6: Continued.

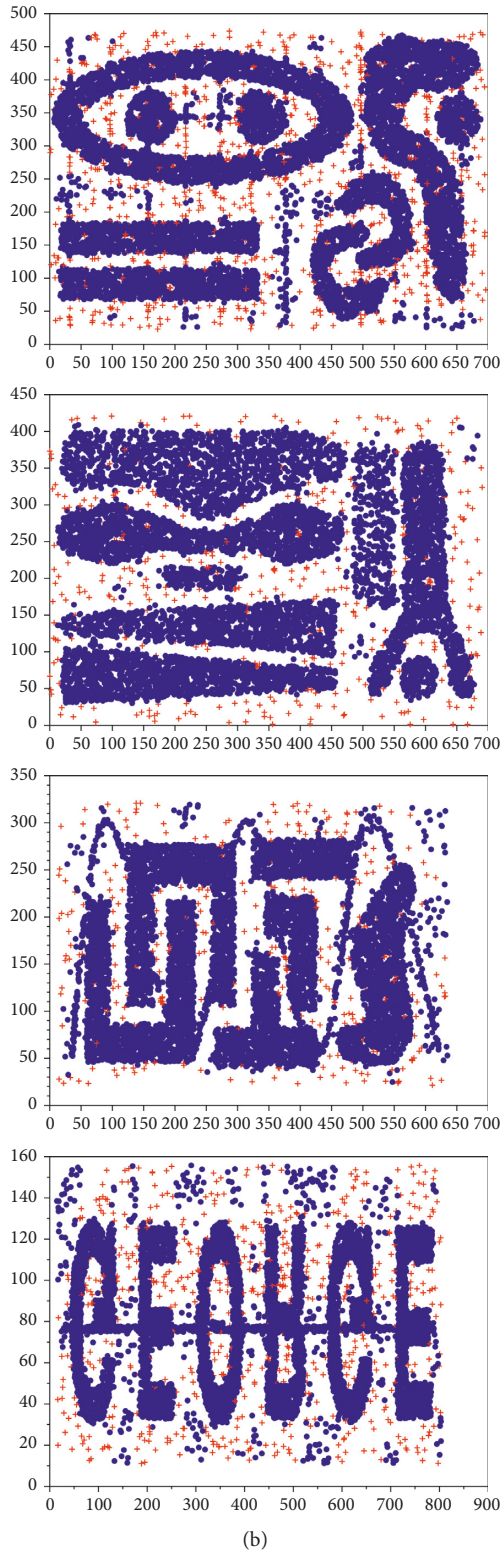


Figure 6: Results of LOF (a) and FLDS (b) on some 2D data sets: the outliers marked as red plus.

these results is illustrated in Table 5. The total accuracy obtained 98.9% compared with 96.26% in the paper [14]. We can explain the results obtained by IncrementalSSGC by the fact that the algorithm used the distance based on shared nearest

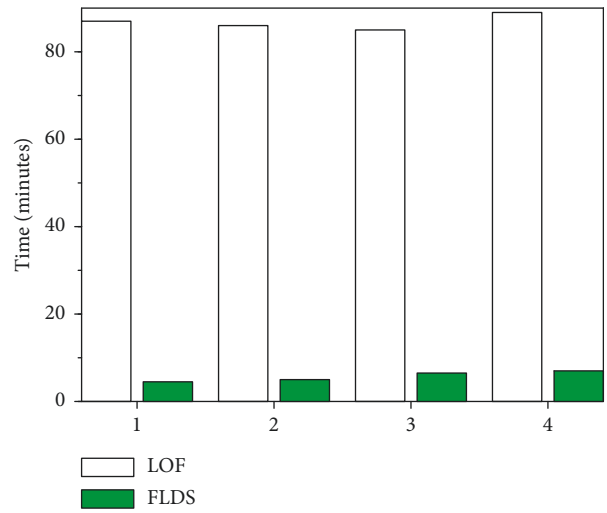


Figure 7: Running time comparison between FLDS and LOF for four 2D data sets.

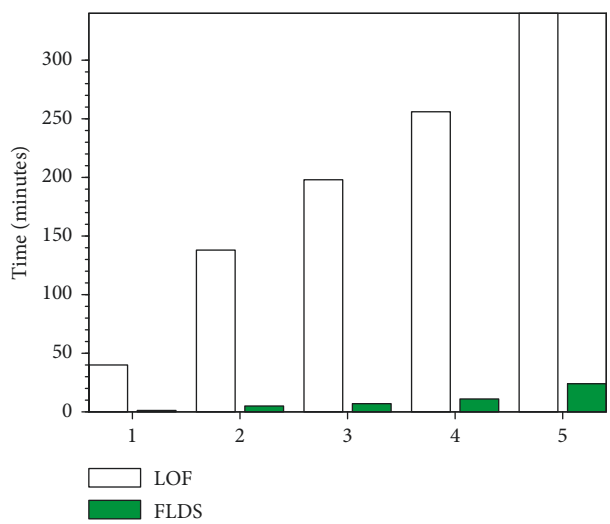


Figure 8: Running time comparison between FLDS and LOF for five AWID data sets.

neighbors, which overcome the limit of transitional distance measures such as Euclidean or Minskovki distance, and the shared nearest neighbors measure does not depend on the density of data. This proposed system is generally called hybrid method which is one of the best strategies in developing Intrusion Detection Systems [7, 9] in which there is no single classifier that can exactly detect all kinds of classes.

We also note that for real applications, whenever an attack appears, the system needs to immediately produce a warning. The multistage system-based machine learning techniques provide a solution for users for constructing the real IDS/IPS system that is one of the most important problems in the network security.

5. Conclusion

This paper introduces an incremental semisupervised graph-based clustering and a fast outlier detection

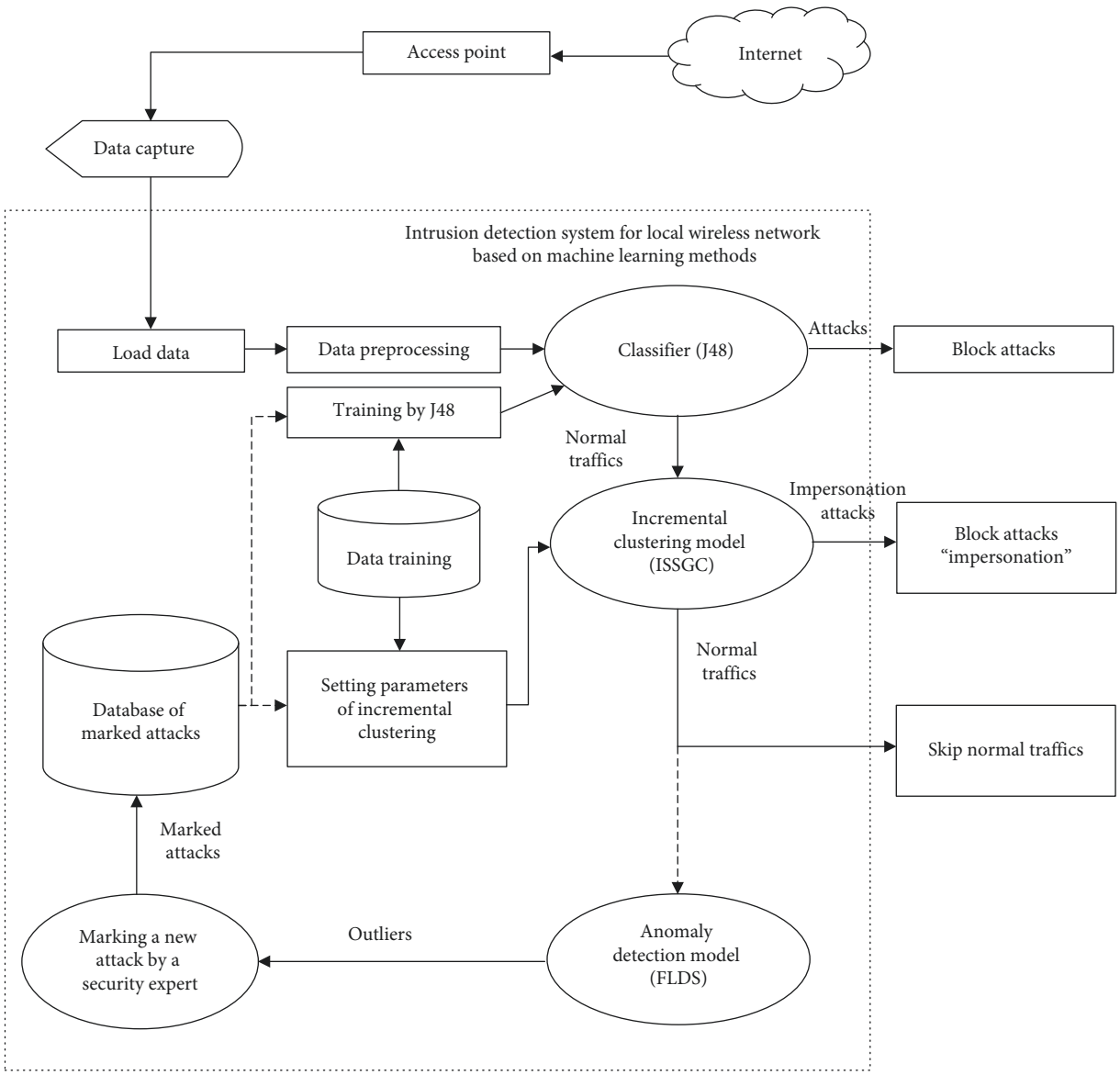


Figure 9: A new framework for intrusion detection in 802.11 networks.

Table 5: Confusion matrix for AWID data set using J48 and IncrementalSSGC in proposed framework.

Normal	Flooding	Impersonation	Injection	Classification
530588	116	6	75	Normal
2553	5544	0	0	Flooding
2	0	16680	0	Injection
3297	148	0	16364	Impersonation

method. Both methods can be used in a hybrid framework for the intrusion detection problem of WiFi data sets (AWID). Our proposed multistage system-based machine learning techniques provide a solution to guideline for constructing the real IDS/IPS system that is one of the most important problems in the network security. Experiments conducted on the extracted data sets from the AWID and UCI show the effectiveness of our proposed methods. In the near future, we will continue to develop other kinds of machine learning methods for intrusion detection problem and test for other experimental setup.

Data Availability

The data used to support the findings of this study can be downloaded from the AWID repository (http://icsdweb.aegean.gr/awid/download.html).

Conflicts of Interest

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

References

[1] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*, MIT press, Cambridge, MA, USA, 2012.

[2] B. Ahmad, W. Jian, and Z. Anwar Ali, "Role of machine learning and data mining in internet security: standing state with future directions," *Journal of Computer Networks and Communications*, vol. 2018, Article ID 6383145, 10 pages, 2018.

[3] T. Bakhshi and B. Ghita, "On internet traffic classification: a two-phased machine learning approach," *Journal of Computer Networks and Communications*, vol. 2016, Article ID 2048302, 21 pages, 2016.

[4] B. Luo and J. Xia, "A novel intrusion detection system based on feature generation with visualization strategy," *Expert Systems with Applications*, vol. 41, no. 9, pp. 4139–4147, 2014.

[5] W.-C. Lin, S.-W. Ke, and C.-F. Tsai, "CANN: an intrusion detection system based on combining cluster centers and nearest neighbors," *Knowledge-Based Systems*, vol. 78, pp. 13–21, 2015.

[6] C.-F. Tsai and C.-Y. Lin, "A triangle area based nearest neighbors approach to intrusion detection," *Pattern Recognition*, vol. 43, no. 1, pp. 222–229, 2010.

[7] F. Kuang, S. Zhang, Z. Jin, and W. Xu, "A novel SVM by combining kernel principal component analysis and improved chaotic particle swarm optimization for intrusion detection," *Soft Computing*, vol. 19, no. 5, pp. 1187–1199, 2015.

[8] M. E. Aminanto and K. Kim, "Detecting impersonation attack in WiFi networks using deep learning approach," in *Information Security Applications*, D. Choi and S. Guilley, Eds., Vol. 10144, Springer, Berlin, Germany, 2016.

[9] A. A. Aburomman and M. B.I. Reaz, "A novel SVM-kNN-PSO ensemble method for intrusion detection system," *Applied Soft Computing*, vol. 38, pp. 360–372, 2016.

[10] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: identifying density-based local outliers," in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pp. 93–104, Dallas, TX, USA, May 2000.

[11] V. Hautamäki, I. Kärkkäinen, and P. Fränti, "Outlier detection using *k*-nearest neighbour graph," in *Proceedings of the 17th International Conference on Pattern Recognition*, pp. 430–433, Cambridge, MA, USA, August 2004.

[12] V. Thang and F. F. Pashchenko, "A new incremental semi-supervised graph based clustering," in *Proceedings of the IEEE International Conference on Engineering and Telecommunication*, Moscow, Russia, March 2018.

[13] V. Thang, D. V. Pantiukhin, and A. N. Nazarov, "FLDS: fast outlier detection based on local density score," in *Proceedings of the IEEE International Conference on Engineering and Telecommunication*, pp. 137–141, Moscow, Russia, November 2016.

[14] C. Kolias, G. Kambourakis, A. Stavrou, and S. Gritzalis, "Intrusion detection in 802.11 networks: empirical evaluation of threats and a public dataset," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 184–208, 2016.

[15] Ch. Gupta and R. Grossman, "GenIc:a single pass generalized incremental algorithm for clustering," in *Proceedings of the Fourth SIAM International Conference on Data Mining*, pp. 147–153, Lake Buena Vista, FL, USA, April 2004.

[16] M. Ester, H.-P. Kriegel, J. Sander, M. Wimmer, and X. Xu, "Incremental clustering for mining in a data warehousing environment," in *Proceedings of the International Conference on Very Large Data Bases*, pp. 323–333, New York, NY, USA, August 1998.

[17] V. Chandrasekhar, C. Tan, M. Wu, L. Li, X. Li, and J.-H. Lim, "Incremental graph clustering for efficient retrieval from streaming egocentric video data," in *Proceedings of the International Conference on Pattern Recognition*, pp. 2631–2636, Stockholm, Sweden, August 2014.

[18] A. M. Bagirov, J. Ugon, and D. Webb, "Fast modified global K-means algorithm for incremental cluster construction," *Pattern Recognition*, vol. 44, no. 4, pp. 866–876, 2011.

[19] A. Bryant, D. E. Tamir, N. D. Rishe, and K. Abraham, "Dynamic incremental fuzzy C-means clustering," in *Proceedings of the Sixth International Conference on Pervasive Patterns and Applications*, Venice, Italy, May 2014.

[20] Z. Yu, P. Luo, J. You et al., "Incremental semi-supervised clustering ensemble for high dimensional data clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 3, pp. 701–714, 2016.

[21] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: a survey," *ACM Computing Surveys*, vol. 41, no. 3, pp. 1–58, 2009.

[22] J. Tang, Z. Chen, A. Fu, and D. Cheung, "Enhancing effectiveness of outlier detections for low density patterns," in *Proceedings of the Sixth Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, Taipei, Taiwan, May 2002.

[23] S. Papadimitriou, H. Kitagawa, P. B. Gibbons, and C. Faloutsos, "Loci: fast outlier detection using the local correlation integral," in *Proceedings of the 19th International Conference on Data Engineering*, pp. 315–326, Bangalore, India, March 2003.

[24] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 226–231, Portland, OR, USA, August 1996.

[25] L. Ertöz, M. Steinbach, and V. Kumar, "Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data," in *Proceedings of the SIAM International Conference on Data Mining*, pp. 47–58, San Francisco, CA, USA, May 2003.

[26] E. M. Knorr, R. T. Ng, and V. Tucakov, "Distance-based outliers: algorithms and applications," *The International Journal on Very Large Data Bases*, vol. 8, no. 3-4, pp. 237–253, 2000.

[27] S. Basu, I. Davidson, and K. L. Wagstaff, "Constrained clustering: advances in algorithms, theory, and applications," in *Chapman and Hall/CRC Data Mining and Knowledge Discovery Series*, , CRC Press, Boca Raton, FL, USA, 1st edition, 2008.

[28] A. A. Abin, "Clustering with side information: further efforts to improve efficiency," *Pattern Recognition Letters*, vol. 84, pp. 252–258, 2016.

[29] Y. Shi, C. Otto, and A. K. Jain, "Face clustering: representation and pairwise constraints," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 7, pp. 1626–1640, 2018.

[30] A. A. Abin and B. Hamid, "Active constrained fuzzy clustering: a multiple kernels learning approach," *Pattern Recognition*, vol. 48, no. 3, pp. 953–967, 2015.

[31] S. Xiong, J. Azimi, and X. Z. Fern, "Active learning of constraints for semi-supervised clustering," *IEEE Transactions on*

Knowledge and Data Engineering, vol. 26, no. 1, pp. 43–54, 2014.

[32] K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl, "Constrained K-means clustering with background knowledge," in *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 577–584, Williamstown, MA, USA, June 2001.

[33] I. Davidson and S. S. Ravi, "Using instance-level constraints in agglomerative hierarchical clustering: theoretical and empirical results," *Data Mining and Knowledge Discovery*, vol. 18, no. 2, pp. 257–282, 2009.

[34] B. Kulis, S. Basu, I. Dhillon, and R. Mooney, "Semi-supervised graph clustering: a kernel approach," *Machine Learning*, vol. 74, no. 1, pp. 1–22, 2009.

[35] V.-V. Vu, "An efficient semi-supervised graph based clustering," *Intelligent Data Analysis*, vol. 22, no. 2, pp. 297–307, 2018.

[36] X. Wang, B. Qian, and I. Davidson, "On constrained spectral clustering and its applications," *Data Mining and Knowledge Discovery*, vol. 28, no. 1, pp. 1–30, 2014.

[37] D. Mavroeidis, "Accelerating spectral clustering with partial supervision," *Data Mining and Knowledge Discovery*, vol. 21, no. 2, pp. 241–258, 2010.

[38] L. Lelis and J. Sander, "Semi-supervised density-based clustering," in *Proceeding of IEEE International Conference on Data Mining*, pp. 842–847, Miami, FL, USA, December 2009.

[39] D.-D. Le and S. Satoh, "Unsupervised face annotation by mining the web," in *Proceedings of the 8th IEEE International Conference on Data Mining*, pp. 383–392, Pisa, Italy, December 2008.

[40] D. Yan, L. Huang, and M. I. Jordan, "Fast approximate spectral clustering," in *Proceedings of the Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 907–916, New York, NY, USA, June 2009.

[41] C. Zhong, M. Malinen, D. Miao, and P. Fränti, "A fast minimum spanning tree algorithm based on K-means," *Information Sciences*, vol. 295, pp. 1–17, 2015.

[42] V. Chaoji, M. Al Hasan, S. Salem, and M. J. Zaki, "SPARCL: an effective and efficient algorithm for mining arbitrary shape-based clusters," *Knowledge and Information Systems*, vol. 21, no. 2, pp. 201–229, 2009.

[43] A. Asuncion and D. J. Newman, *UCI Machine Learning Repository*, American Statistical Association, Boston, MA, USA, 2015, http://archive.ics.uci.edu/ml/index.php.

[44] G. Karypis, E.-H. Han, and V. Kumar, "Chameleon: hierarchical clustering using dynamic modeling," *Computer*, vol. 32, no. 8, pp. 68–75, 1999.

Hindawi
Journal of Computer Networks and Communications
Volume 2019, Article ID 8012568, 12 pages
https://doi.org/10.1155/2019/8012568

Research Article

Advanced Support Vector Machine- (ASVM-) Based Detection for Distributed Denial of Service (DDoS) Attack on Software Defined Networking (SDN)

Myo Myint Oo, Sinchai Kamolphiwong, Thossaporn Kamolphiwong, and Sangsuree Vasupongayya

Department of Computer Engineering, Faculty of Engineering, Prince of Songkla University (Hatyai Campus), Hatyai, Songkhla 90110, Thailand

Correspondence should be addressed to Thossaporn Kamolphiwong; kthossaporn@coe.psu.ac.th

Received 10 December 2018; Accepted 4 February 2019; Published 4 March 2019

Guest Editor: Mazdak Zamani

Software Defined Networking (SDN) has many advantages over a traditional network. The great advantage of SDN is that the network control is physically separated from forwarding devices. SDN can solve many security issues of a legacy network. Nevertheless, SDN has many security vulnerabilities. The biggest issue of SDN vulnerabilities is Distributed Denial of Service (DDoS) attack. The DDoS attack on SDN becomes an important problem, and varieties of methods had been applied for detection and mitigation purposes. The objectives of this paper are to propose a detection method of DDoS attacks by using SDN based technique that will disturb the legitimate user's activities at the minimum and to propose Advanced Support Vector Machine (ASVM) technique as an enhancement of existing Support Vector Machine (SVM) algorithm to detect DDoS attacks. ASVM technique is a multiclass classification method consisting of three classes. In this paper, we can successfully detect two types of flooding-based DDoS attacks. Our detection technique can reduce the training time as well as the testing time by using two key features, namely, the volumetric and the asymmetric features. We evaluate the results by measuring a false alarm rate, a detection rate, and accuracy. The detection accuracy of our detection technique is approximately 97% with the fastest training time and testing time.

1. Introduction

Nowadays, networking technologies are gradually developed for advanced infrastructure. With the development of advanced technologies, the explosion of mobile devices, server virtualization techniques, and cloud services are the strongest points in a traditional network architecture. Most traditional network architectures are hierarchical arrangement in a client-server model. Today's applications access different databases and servers in different network domains. Therefore, multiple clients and multiple server cases are expected. Thus, the traffic patterns may not be the same. Enterprise businesses' public and private cloud services want to provide the agility to access applications, infrastructures, and other IT resources on demand. This can be solved by using Software Defined Networking (SDN) to provide a network infrastructure. SDN becomes an important role in overcoming the limitations of a traditional networking. The most obvious thing in SDN is decoupling of the data plane and the control plane. The control plane is the plane that determines where to send the traffic, and the data plane is the plane that executes this decision and actually forwards the traffic. Although SDN has many advantages, some challenging issues that need to be solved still exist. One of the big challenging issues is the SDN security issue. There are many kinds of network attacks on SDN. Among them, Distributed Denial of Service (DDoS) attack is very well known and has the highest impact on SDN [1]. There are varieties of researches for detection of the DDoS attack on SDN network [2].

In this paper, we propose Advanced Support Vector Machine (ASVM) technique as an enhancement of an existing Support Vector Machine (SVM) algorithm to detect DDoS attacks. We have explored three research problems with our proposed technique [3]. The first problem is the extension of the multiclass problem in the Support Vector Machine (SVM) algorithm. If the SVM algorithm is applied in a DDoS attack detection problem on a SDN network, some of the network traffic attributes are multivalue attributes. However, the SVM is originally designed for a binary classification. Therefore, multiclass classification is a big problem for applying SVM. The second problem is the long training and testing time required for the SVM algorithm. The SVM classifier gives a low false-positive rate and a high classification accuracy. However, the SVM algorithm takes more time to train and test for the detection of the attack. The third problem is the efficiency of the SDN enabled centralized network. In previous proposed SDN architectures, the network system used only a single controller. Therefore, using multiple controllers is the most important issue for our proposed network infrastructure. Our contributions are summarized as follows.

We create test cases of the proposed model by using Miniedit and OpenDaylight controllers [4]. In the traffic generation process, we generate normal traffics, UDP flooding DDoS attack traffics [5], and SYN flooding DDoS attack traffics [6]. In the traffic collection process, we collect the traffic from each switch. In the feature generation process, we generate the volumetric features, average number of packets in a flow, average number of flow bytes and the asymmetric features, amount of packet variations in a flow, the variation of flow bytes, and the average duration of traffics in the sampling interval. In the classification process, we propose the Advance Support Vector Machine (ASVM) method. In the evaluation process, we evaluate the classification result by measuring false alarm rate, detection rate, and accuracy.

The paper is organized as follows: in the second section, we survey a number of related works to our proposed method. In the third section, we discuss the theoretical background used by our research work, Software Defined Networking and Distributed Denial of Service (DDoS) attack. In the fourth section, we present the architecture of our proposed system. In the fifth section, we provide the implementation details of the proposed detection system. In the sixth section, we briefly discuss the experimental results. We discuss the performance evaluation part of our results in the seventh section. Lastly, the eighth section concludes our work and some future works.

2. Related Works

There are two kinds of DDoS detection techniques: signature-based detection and anomaly-based detection techniques. The signature-based detection technique uses the network behaviours. The anomaly-based detection uses the machine-learning techniques. Commonly used machine learning techniques for DDoS attack detection include an artificial neural network (ANN), Support Vector Machine (SVM), Fuzzy Logic, Decision tree, Evolutionary algorithm, Navies Bayes, and k-means clustering algorithms. ANN has been used in the detection of known and unknown DDoS attacks research [7], which shows that we can detect the DDoS attack on the SDN controller with a noticeable accuracy and prevent serious damage to the controller. The perceptron neural network was used in [7], and the evaluation results showed that a significant improvement on the detection rate were achieved while a reduction in false alarm rate is also achieved in comparison with the closest previous work. Furthermore, their system was able to maintain the average detection time at an acceptable level. They would investigate an efficient method to mitigate the attack for the future work. Support Vector Machine (SVM) is used to classify the DDoS attack with normal traffic because of its high accuracy and less false-positive rate in [8]. SVM classifier was compared with other classifiers for detection of the DDoS attack and SVM provided an accurate classification than other techniques. DDoS real-time detection and the integration of the traffic pattern built in SVM with SDN controller were their future work. Fuzzy Logic can be used for real traffic detection of the DDoS attack on SDN [2]. The authors have solved the existing problems of the OpenFlow protocol. They proposed Fuzzy Logic-based DDoS mitigation algorithm that deployed multiple criterion for DDoS detection. Their system demonstrated the ability to detect and filter 97% of the attack flows with a false-positive rate of 5%. They would like to extend the OpenFlow protocol to achieve robust and faster performance.

Moreover, the researchers have designed the system to detect DDoS attacks based on a decision-tree technique, and they traced back to the approximate locations of the attacker with a traffic flow pattern-matching technique [9]. Their system could detect the attack with the false-positive ratio of 1.2%–2.4%. They conducted their experiment on the DETER system. Their results indicated that their proposed system was capable of detecting the attacks and tracing back with a high accuracy. Evolutionary algorithms (EAs) for detecting DDoS attack are presented in SDN [10]. The researchers reviewed four types of EAs that widely applied in current SDNs: Genetic algorithms (GAs), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and Simulated Annealing (SA). All four EAs were compared, and the applications of these four EAs in SDNs were categorized. In order to get a good DDoS detection technique, the researchers have provided a better solution of detections using a features analysis [11]. Naive Bayes classifier algorithm was used in order to classify the packets into normal and attack packets. The use of information gain algorithm increases the performance. CAIDA 2008 and CAIDA anonymous trace 2015 datasets were used for their feature selection and classification. A method to detect a DoS attack using clustering technique with the k-means algorithm that available to be modified and developed in many possible ways was used in [12]. By using this algorithm, their result was evaluated on detection rate, accuracy, and false-positive rate. Their method has been evaluated by using DARPA 98 dataset with the satisfying

result. In the future, they would like to improve in minimize false-positive rate.

3. Software Defined Networking (SDN)

Software Defined Networking (SDN) is an emergent network architecture where the network control is dynamic, manageable, adaptable, and physically separated from forwarding devices [13]. There are three layers in a SDN architecture, including the infrastructure layer (Data plane), the Control layer (Control Plane), and the Application layer (Management plane), as shown in Figure 1.

The first layer, the infrastructure layer is composed of switches. The major work of these switches is forwarding the incoming packets according to the flow tables. Forwarding decisions can be decided and configured by the control plane through the southbound protocol. The first standard of the southbound protocol is the OpenFlow protocol [14]. OpenFlow is defined from the OpenFlow switch specification published by Open Network Foundation (ONF). The second layer, the control layer, maintains a centralized view of the network and open interfaces. This layer allows applications to control the underlying networking. This layer also provides the interconnection of applications on the top and the bottom of the architecture. The third layer, the application layer is composed of applications managing and securing the underlying network. The application could be running on the controller or the application could communicate through the northbound Application Programming Interface (API) of the controller. There is no standard API for the northbound protocol. The main idea of SDN architecture is the separation of the data plane and the control plane. This network separation has many benefits in terms of the network flexibility and controllability.

Although SDN has many advantages over the traditional network, it faces some challenges. The main challenges of SDN are reliability, scalability, security, and interoperability. Among the challenges, we emphasize on the security of SDN. Each plane of SDN has vulnerabilities. In the data plane, single network devices, switches are quite vulnerable to different kind of attacks such as Denial of Service (DoS) attack, Distributed Denial of Service (DDoS) attack, data modification, repudiation, blackhole attack, and side channel attack. DoS and DDoS are the most popular attacks on the data plane so that the network cannot be accessed by the legitimate users. In the control plane, the controller is the easiest target of DDoS because the first packet of each flow must be sent to the controller, and sometimes it can cause a bottleneck condition. Moreover, some malicious attacks, DoS, blackhole, and fake flow rule generation can also occur at the control plane. In the application plane, there is some vulnerability concerning the DDoS attack, for example, in Smart City application [15]. Good features of SDN offer new opportunities to defeat attacks in cloud computing environments. DoS, DDoS flooding attacks, are the main methods to destroy the availability of cloud computing [16]. Therefore, many researchers propose solutions and countermeasures of DDoS attacks on a SDN network.

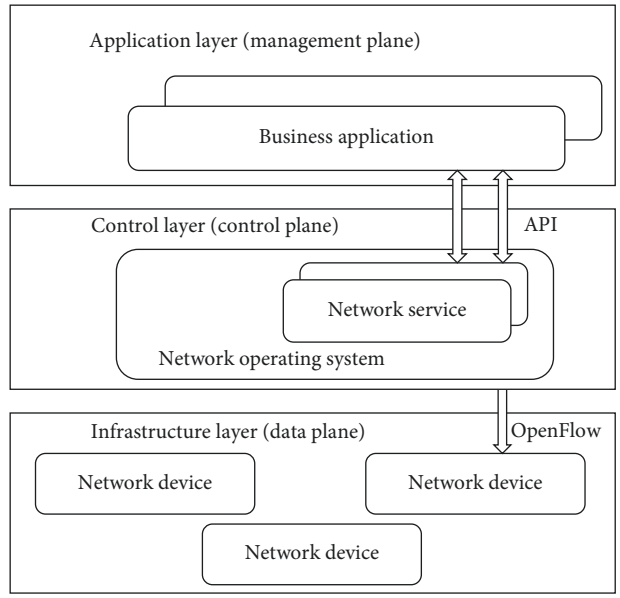


Figure 1: The structure of Software Defined Networking (SDN).

4. Distributed Denial of Service (DDoS) Attack

Distributed Denial of Service (DDoS) attack is a kind of DoS attack that the bombardment of simultaneous data is accessing to the server to hide the availability of resources in the network. According to the state of the internet security, summer 2018 report [17], the largest DDoS attack with a record peak 1.35 Tbps was observed on Wednesday, February 28, 2018. In this kind of DDoS attack, the attackers did not use any botnet network. They use weaponized misconfigured Memcached servers to conduct the DDoS attack. The attack size is more than twice that of Mirai botnet DDoS attack in 2016. This attack originated from thousand autonomous systems (ASNs) across tens of thousands of endpoints. It was an amplification attack using the Memcached-based approach producing 126.9 million packets per second [18]. The DDoS attack can be classified into three basic categories: volume-based attacks, protocol attacks, and application attacks. Under a volume-based attack, the target can be flooded with heavy traffics in order to exhaust its bandwidth. This type of attack can be detected by byte per second. Flooding attacks, User Datagram Protocol (UDP) flood, and Internet Control Message Protocol (ICMP) flood are volume-based attacks. In this research, we have been analyzed volume-based attacks. Under a protocol attack, the resources can be exhausted by exploiting the network protocol. The result of this attack is the unavailable underlying operating system. This type of attack can be detected by packets per second. SYN flood, ping of death, and smurf attacks are protocol attacks. Under an application attack, the application or server can be crashed by exploiting the application layer protocol. This attack can be detected by request per second. Hypertext Transfer Protocol (HTTP) flooding and Slowloris are application attacks [19].

The most common type of DDoS attacks include SYN flooding attack, UDP flooding, ICMP flooding, HTP

flooding, ping of death attack, smurf attack, and slowloris attack. The details of each attack are as follow. SYN flooding attack can exploit the weakness of TCP connection sequence, three-way handshake [20]. At first, the host machine receives a synchronized (SYN) message to start the "handshake." The server acknowledges the message by sending an acknowledge (ACK) flag to the first host and then closes the connection. Under a SYN flooding attack, the spoofed messages are sent and the connection does not close, and the service can be shutting down. UDP flooding attack can exploit the session less User Datagram Protocol (UDP) [21]. At first, the attackers send a large amount of UDP packets to random ports on the target, and the target host checks for applications on that port. No listening application on that port is found, so it replies with ICMP destination unreachable packet. This attack can consume more resources even though the host is unreachable. ICMP flooding attack can exploit by consuming a large number of ICMP pings [22]. Under an ICMP attack, ICMP echo packets are frequently sent without waiting for any echo reply, and the target attempts to reply these ICMP echo requests. Therefore, its outgoing bandwidth can be affected. HTTP flooding attack can be exploited by using legitimate GET or POST requests [23]. Although this attack uses less bandwidth than other kinds of DDoS attacks, it can force the server to use its maximum resources. Ping of death attack can exploit IP protocols by sending malicious pings to the system [24]. This attack does not require huge data to bring down the victim; it only needs to exploit the standard protocol. Smurf attack can exploit IP and ICMP protocol by using a malware program called smurf [25]. This attack spoofs an IP address and pings these addresses on a given network using smurf. Slowloris attack can break down the server by having maximum connections with attackers [26]. At first, attackers send partial HTTP requests to the server. The server keeps the connection for these requests, and the result is DoS to legitimate requests.

5. Detection of DDoS Attack on SDN by Using Advanced Support Vector Machine (ASVM)

Under our proposed framework, the DDoS attack will be detected on the SDN network by using the Advanced Support Vector Machine (ASVM) method. The proposed research presents a customizable DDoS defence framework which generates DDoS attack alerts by considering the application's security requirements [1]. Our proposed framework has been motivated by the concept that different applications have different security requirements. From our proposed framework, a DDoS attack detection solution must include a customizable reaction mechanism for generating DDoS attack alerts. Our proposed system leverages the programming and dynamic nature of SDN and implements an adaptive DDoS protection mechanism. Figure 2 illustrates the architecture of the proposed framework.

Attackers or normal users have been sent the packets to the OpenFlow Switches. When the packet arrives at the OpenFlow switch, the packet information will be checked such as the information on the packet header fields including source port, destination port, source IP address, and destination IP address. The information of the incoming packets will be checked against the flow entries, if a match is found then a specified action can be executed. Otherwise, the packet will be sent to the OpenDaylight controller via the southbound API using a packet_in control message. Controllers are connected as a cluster. When the traffics arrived at the OpenDaylight controller cluster, they will be forwarded via the northbound API to the Detection of DDoS attack by ASVM of application layer. The packet will be classified as a DDoS attack traffic or a normal traffic. The components of our proposed framework consist of four modules including the traffic generation, the traffic data collection, the feature extraction, and the classification of attack or normal by ASVM method. Two kinds of flooding-based DDoS attacks and normal traffics are generated. We have collected the traffic data from each OpenFlow switche. The five features have been extracted and classified as DDoS attacks or normal traffics by ASVM method. The graphical representation of these modules can be seen in Figure 3.

6. Traffic Generation

The generation of two DDoS attack traffics and normal traffics is implemented in this work. Two DDoS attacks are UDP flooding attacks and SYN flooding attacks. UDP flooding attack is a type of Denial of Service (DoS) attack in which the random ports on the target's host will be flooded with IP packets using User Datagram Protocol (UDP). Under a UDP flooding attack, first, the victim's IP addresses are determined; then the source port and the destination port are initialized to 80 and 1. Each time, 2000 packets are generated. The packets interarrival time for UDP attack traffics is 0.03 seconds. Scapy, a packet generation tool for computer networks written in python language, is used for generating the packets in this work. For each random source IP address, a packet is created with the source IP and the destination IP using scapy. Scapy can forge or decode packets; Scapy can send the packets on the wire; Scapy can capture the packets; and Scapy can match the requests and the replies. Scapy can also handle tasks like scanning, tracerouting, probing, unit tests, attacks, and network discovery. After the packet is created, it must be sent to the destination IP address within the time interval. The step by step process of the UDP flooding attack on the SDN network can be seen in Figure 4.

SYN flooding attack is a type of DoS attack that exploits the normal three-way handshake procedure to consume the resources on the targeted server and render it unresponsive by using the TCP connection. Under a SYN flooding attack, the victim IP addresses, the victim Port, and the number of packets must be determined. Then, an IP packet with a random source IP and the victim IP will be generated. We also need to create the TCP packet with a random source port, the victim port, 's' flag, packet sequence, and time window. At last, both the IP and TCP

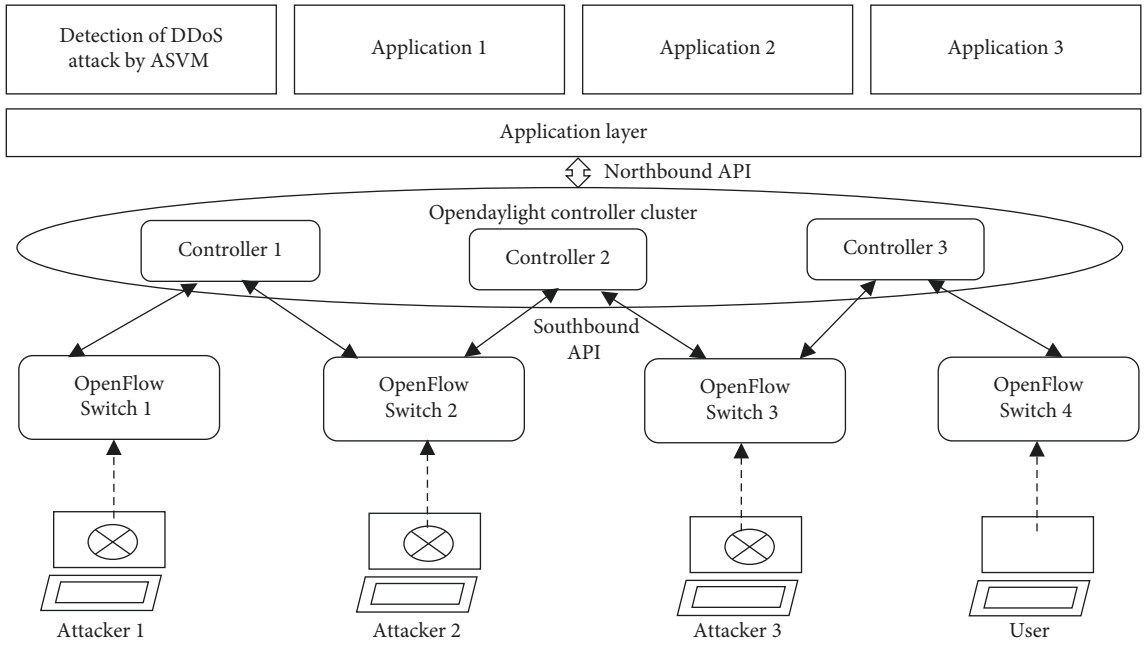


Figure 2: The proposed SDN-based DDoS attack detection framework.

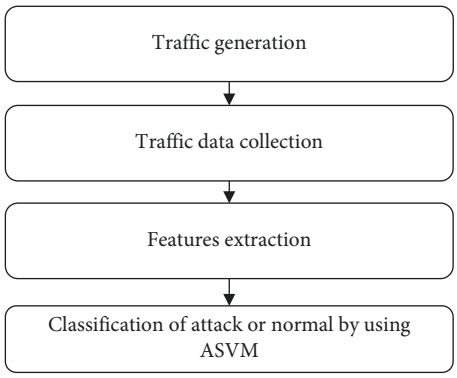


Figure 3: Four modules of our proposed system framework.

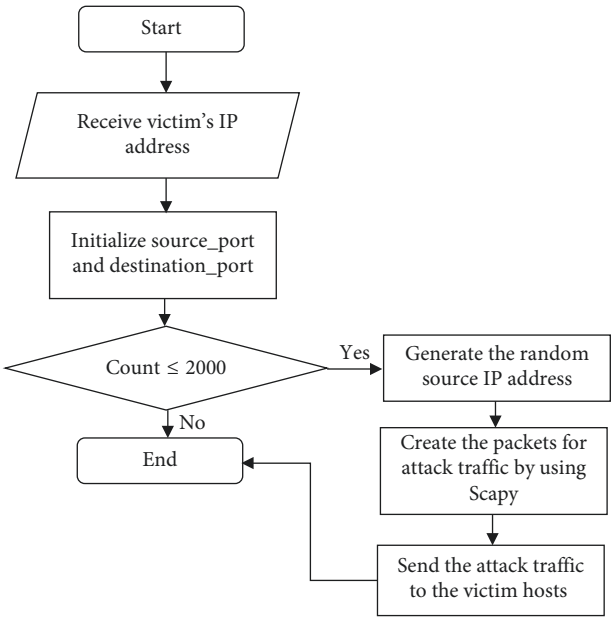


Figure 4: Step by step process of UDP Flooding Attack.

packets will be sent to the victim host. The step by step process of a SYN flooding attack on the SDN network can be seen in Figure 5. The normal traffics are also generated as shown in Figure 6. For a normal traffic generation, the last number of host's destination IP address must be determined. Each time, 1000 packets are generated because the average number of packets at a normal condition is approximately 1000 packets. The packets interarrival time for normal traffic generation is 0.1 second. The random source IP address is used each time. Scapy is also used for creating the normal traffic packets to be sent to the destination host.

7. Traffic Data Collection

For the detection of a DDoS attack on a SDN network, the traffic data collection is the main part of the system. We can collect the traffic data information through the OpenFlow protocol from the OpenFlow switches. In SDN, the traffic data are stored in the flow table within the OpenFlow switches. When we want to extract the traffic data, the OpenFlow switch responds to the onp_-flow_stats_requst message and periodically sends this request message to the controller. OpenDaylight controller is used in our research to manage and control the data-obtaining period and flow-deleting period within the time interval. We can send the flow request command, "sudo ovs-ofctl dump-flows s1" to each switch in order to collect the traffic flow information of the flow table. An example of the extracted traffic flow information from a switch is shown in Figure 7.

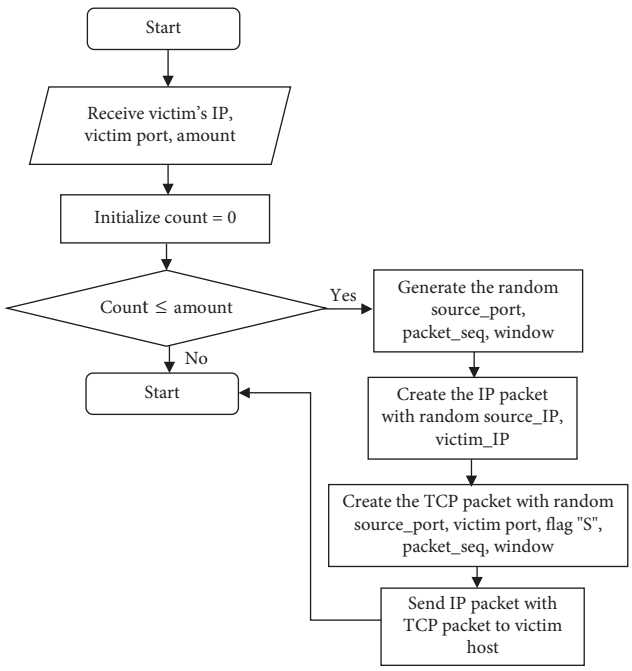


Figure 5: Step by step process of SYN Flooding Attack.

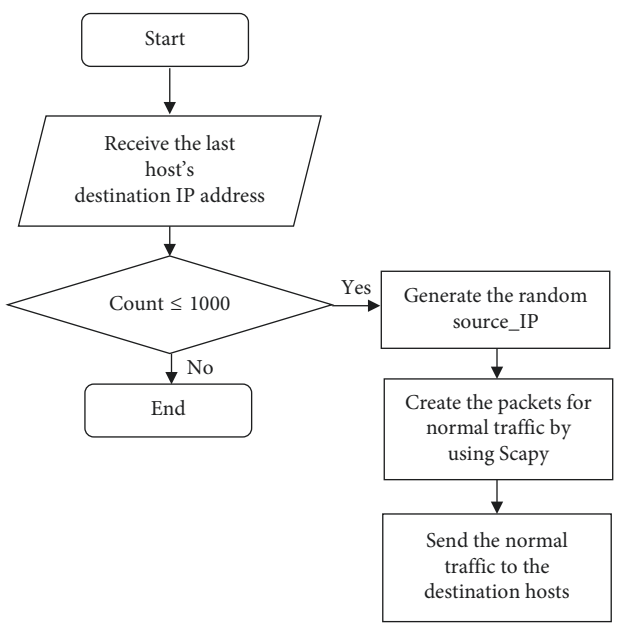


Figure 6: Step by step process of normal traffic generation.

8. Feature Extraction

After collecting the traffic data, the next step is the extraction of traffic features. The collected malicious traffic flows on the SDN network can be analyzed by inspecting various characteristic values of the flow table. When the traffic data from the switch is extracted, we can collect the number of packets that the host is sending, the number of bytes that the host is used, and the duration that takes for sending a packet to or receiving a packet from other hosts. The nature of the SYN and UDP flooding attack traffics are in a form of normal distribution [27]. For volumetric and asymmetric nature of the traffic patterns, there are five different kinds of traffic features to be analyzed, including average number of flow packets in the sampling interval (ANPI), average number of flow bytes in the sampling interval (ANBI), variation of flow packets in the sampling interval (VPI), variation of flow bytes in the sampling interval (VBI), and average duration of traffics in the sampling interval (ADTI).

ANPI is the sum of the number of flow packets in each flow per total flows at the sampling interval as shown in Equation (1). ANPI is used for a detection of the DDoS attack on the SDN network because the nature of the DDoS attack is sending a large number of packets in order to disable the controller. Therefore, we can detect a malicious traffic by measuring the number of flow packets.

$$\text{ANPI} = \frac{\sum_{i=1}^{\text{total flows}} \text{flow packet}_i}{\text{total flows}}. \tag{1}$$

ANBI is the sum of the number of flow bytes in each flow per total flows at the sampling interval as shown in Equation (2). ANBI is used for a detection of the DDoS attack on the SDN network because most DDoS attackers want to send the packet; they do not consider the data bytes of the packets. Thus, the flow byte measurement can indicate a malicious traffic.

$$\text{ANBI} = \frac{\sum_{i=1}^{\text{total flows}} \text{flow byte}_i}{\text{total flows}}. \tag{2}$$

VPI is the measurement of the standard deviation of the number of flow packets at sampling interval as shown in Equation (3). We can detect the DDoS attack on SDN network by considering the VPI feature because most DDoS attackers randomly create the packets in order to send to the hosts; they do not consider the full data packet, and mostly empty packets are used.

$$\text{VPI} = \sqrt{\frac{\sum_{i=1}^{\text{total flows}} (\text{flow packet}_i - \text{ANPI})^2}{\text{total flows}}}. \tag{3}$$

VBI is the measurement of the standard deviation of the number of flow bytes at the sampling interval as shown in Equation (4). We can use the VBI feature in the detection of the DDoS attack on the SDN network because most DDoS attackers do not consider the flow bytes of the packets. Therefore, we can detect malicious traffic by measuring the variation of the flow bytes.

$$\text{VBI} = \sqrt{\frac{\sum_{i=1}^{\text{total flows}} (\text{flow packet}_i - \text{ANBI})^2}{\text{total flows}}}. \tag{4}$$

ADTI is the sum of each duration of the SDN traffic per a sampling interval as shown in Equation (5). We can detect a malicious traffic nature by measuring the ADTI feature of the SDN traffic. The DDoS attackers send a large number of packets within a certain interval. Therefore, we can measure

```
mininet> sh ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
cookie=0x0, duration=165.700s, table=0, n_packets=8, n_bytes=560, idle_age=20,
in_port=1 actions=FLOOD
```

Figure 7: An example of the traffic flow information from a switch.

the malicious traffic behaviours at each duration of the packet sending and receiving period.

$$\text{ADTI} = \frac{\text{each duration of SDN traffic}}{\text{sampling interval}}. \quad (5)$$

9. ASVM Classification of Attack or Normal Traffic

In our proposed system, the ASVM method is utilized to classify each packet to be attack or normal traffic. The ASVM method is the advanced Support Vector Machine (SVM) algorithm. SVM is a supervised machine learning algorithm that can be used on both classification and regression problems [3]. SVM is widely used in many application areas because of its high accuracy, ability to deal with high-dimensional data, and flexibility in modelling diverse data. SVM is originally used for liner two-class classification problems. In a sample linear two-class classification problem, the assumption is that there are two classes, +1 (positive class) and −1 (negative class). Small letter 'x' denotes a vector with components x_i. The dataset of n points can be shown as

$$D = \{(x_i, y_i)\}_{i=1}^{n}, \quad (6)$$

where x_i denotes the ith characteristic vector in a dataset and y_i is the label associated with x_i. The value of y_i is +1 or −1. The example of linear classification by SVM is shown in Figure 8.

According to Figure 5, there is a straight line separating the vector of class +1 from the vector of class −1. This straight line is denoted as $w \cdot x + b = 0$, where the vector w is called the weight vector and the scalar b is called the bias. The hyperplane of the class label 1 above the straight line is denoted as $w \cdot x + b = 1$ and another hyperplane of the class label −1 below the straight line is denoted as $w \cdot x + b = -1$. When the dataset is linearly separable, this two hyperplanes can be seen as parallel and the distance between them must be as large as possible. The distance between them is calculated as follows:

$$\text{distance between two hyperplanes} = \frac{2}{\|w\|}. \quad (7)$$

Therefore, the distance between the planes must be maximized. As a result, $\|w\|^2/2$ must be minimized. We also need to consider the prevention of the data points from falling into the margin. We need to add the constraint for each "i" either $w \cdot X_i - b \geq 1$ if $y_i = 1$ or $w \cdot X_i - b \leq -1$, if $y_i = -1$. The constraint for each data points need to be lied at the correct side of the margin which is $y_i\,(w \cdot X_i - b) \geq 1$, for

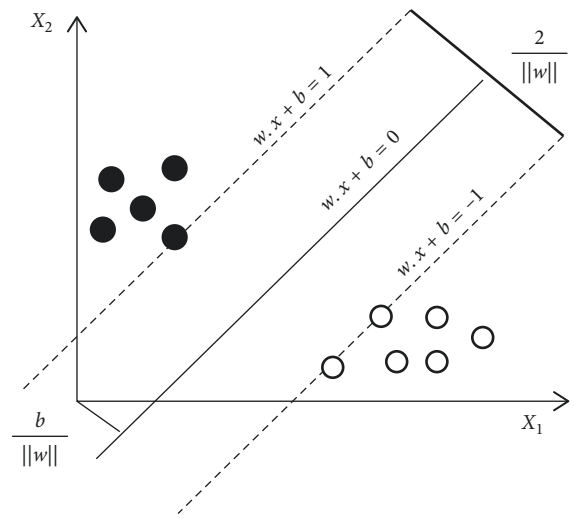


Figure 8: Linear Support Vector Machine (SVM).

all $1 \leq i \leq n$. Therefore, the optimization problem here is minimize $\|w\|^2/2$ subject to $y_i\,(w \cdot X_i - b) \geq 1$, for $i = 1, ., n$. In practice, the data are not linearly separable. There are multiclass. Sometimes, the maximization of margin can cause an error because of a misclassification of the data. In this work, we extend the SVM with Advanced Support Vector Machine (ASVM). We need to consider the slack variables (ξ_i) and the classification error (C). Slack variable is the variable that measures the distance of the point to its marginal hyperplane [28]. The optimal problem is shown in the following equation 8:

$$\begin{aligned} &\text{minimize} \quad \frac{\|w\|^2}{2} + C\sum_{i=1}^{n} \xi_i, \\ &\text{subject to} \quad y_i\left(w \cdot x_i - b\right) \geq 1 - \xi\_i, \quad \xi\_i \geq 0. \end{aligned} \quad (8)$$

The classification error, $C > 0$, gives the relative importance of maximizing the margin and minimizing the amount of slack. In a multiclass classification problem, we need to consider the classifier judgment including one-versus-one and one-versus-some. In one-versus-one, the classification pattern is constructed as $n(n-1)/2$. There are two classes. The sample of the first class is trained as a positive sample and the second class is trained as a negative one. All of these classifiers are needed to classify the data in the testing phases. In one-against-some, the classification pattern is constructed such that each class is trained with the remaining $n-1$ classes. One class of the sample is denoted as positive, and all other samples are denoted as negatives. When we make a decision, it is needed to produce a real-valued confidence

score. When we use the SVM algorithm in the classification problem, the most important thing is choosing the kernel function. Kernel function $K(x\_n, x\_i)$ takes the dataset into a higher dimension space in order to make it possible to separate the data [29]. The kernel function in this work is of the form

$$(x_n \cdot x_i) \longleftarrow K(x_n, x_i) = (\phi(x_n) \cdot \phi(x_i)), \quad (9)$$

where x_n is the support vector data with $n = 1, 2, 3, 4, \ldots, N$. The most useful kernel functions of SVM algorithm are a linear kernel function, Radial Basis Function (RBF), sigmoid, and polynomial. Kernel functions are listed in Table 1.

In this system, we have detected UDP and SYN flooding attacks. Nature of both attacks is normal distribution [30]. In this work, linear kernel and OVS (one-versus-some) decision function are used for classifying the DDoS attack and the normal traffics.

10. Experimental Result and Analysis

The experiments in this work are conducted on the Mininet (version 2.3.0d1) emulator in order to create the SDN network topology on an Ubuntu 16.04 VMware. Our VMware is implemented with 2 processors, 4 MB of RAM, and 20 GB (SCSI) of hard disk. There are the varieties of different controllers: Ryu, ONOS, POX, Floodlight, NOX, and OpenDaylight. Among them, the OpenDaylight (version Beryllium) controller is used for controlling the network topology. OpenDaylight is an open source Java-based SDN controller that is supported by VMware, managed by the Linux Foundation [31]. The OpenDaylight controller has a very large platform with a lot of plugins and features. Mininet is a network emulator that runs the collection of end-hosts, switches, routers, and links on a single Linux kernel, and its results are as same as a real network [32]. Most DDoS attacks use at least three hosts, and the number of hosts can be up to approximately one hundred hosts; at least one switch is used, and the number of controllers used can range from one to as much as possible. Our SDN testbed consists of one hundred hosts (h1 to h100), nine switches (s1 to s9), and three controllers (c0, c1, c2). Four subnets are arranged in our testbed. The experiments are set up on Miniedit. Miniedit is a simple GUI editor for Mininet. Figure 9 shows our implemented testbed.

After running the testbed, the network flows have been added to the nine switches. Open Virtual Switch (OVS) and OpenFlow protocol (version OpenFlow13) are used in our testbed. OVS is a production quality, multilayer virtual switch licensed under the open source Apache 2.0 license [33]. We have been the command, for example, in switch s1 as "sh ovs-ofctl add-flow s1 in-port = 1, action = flood" at our testbed terminal. 126 flows are added for nine switches. In our testbed, each traffic type is generated from 100 scenarios. There are three types of traffics including normal traffics, UDP flooding attacks, and SYN flooding attacks. Under a UDP flooding attack scenario, we use at least five hosts to nine hosts as the attacker hosts and four hosts as the victims.

Table 1: Different kernel function.

No.	Kernel function	Formula
1	Linear	$K(x_n, x_i) = (x_n, x_i)$
2	RBF	$K(x_n, x_i) = \exp(-\gamma \lVert x_n - x_i \rVert^2 + C)$
3	Sigmoid	$K(x_n, x_i) = \tanh(\gamma(x_n, x_i) + r)$
4	Polynomial	$K(x_n, x_i) = (\gamma(x_n, x_i) + r)^d$

Under a SYN flooding attack, four hosts are assigned as attacker hosts and only one victim host. In each scenario, the traffic generation is started first; then the traffic flow information from each switch will be manually collected from each switch. After processing the generation and the collection of traffic data for each scenario, five different traffic features are extracted in order for the ASVM to detect the DDoS attack.

In this experiment, the sampling traffic collection time for attack traffics and normal traffics is 200 seconds. The result of the first feature and ANPI for normal traffics are shown in Figure 10. The trend of the curve has gradually fluctuated within the sampling time. The ANPI feature of attack traffics are shown in Figure 11. During the attack period, the numbers of packets are growing rapidly. The trend of the curve is fluctuated at first, and sometimes, the value reached the highest point depending on the randomly generated attack traffic packets.

The result of the second feature, ANBI in the sampling interval for normal traffics, is shown in Figure 12. The trend of the curve is fluctuated depending on the number of flow bytes for the normal traffics. The value of ANBI for attack traffics within the sampling time is expressed in Figure 13. The attackers send a large number of packets as fast as possible, but they do not consider the data value. Therefore, the ANBI value of attack traffic is regularly from up to down and sometimes apparently reaches the highest point.

The result of the third feature, VPI for normal traffics is shown in Figure 14. Normally, the variation of the flow packets is relatively unchanged. For the attack case, however, the VPI changes rapidly. The VPI curve trend for attack traffics is shown in Figure 15. When the attacks occur within the sampling time, the variation of traffics has fluctuated, and sometimes, it reaches the highest point.

The result of the fourth feature, the normal traffics of VBI, is shown in Figure 16. The trend of the curve is gradually fluctuated, and sometimes, it reaches the lowest points at sampling time 65 and 169 seconds. When the attack occurs in the sampling time, the attackers did not consider the flow byte values of the sending packets. Therefore, the curve trend gradually grows up and down as shown in Figure 17.

The result of the last feature, ADTI for normal traffics and attack traffics, is shown in Figures 18 and 19, respectively. The curve of both types is the same, but the ADTI value of the attack traffics is apparently greater than that of the normal traffics.

The extracted features from the traffic data have been stored as the feature dataset, namely, SDNtrafficDS. The next step is the classification of these dataset by the ASVM method. The classification process is shown in Figure 20.

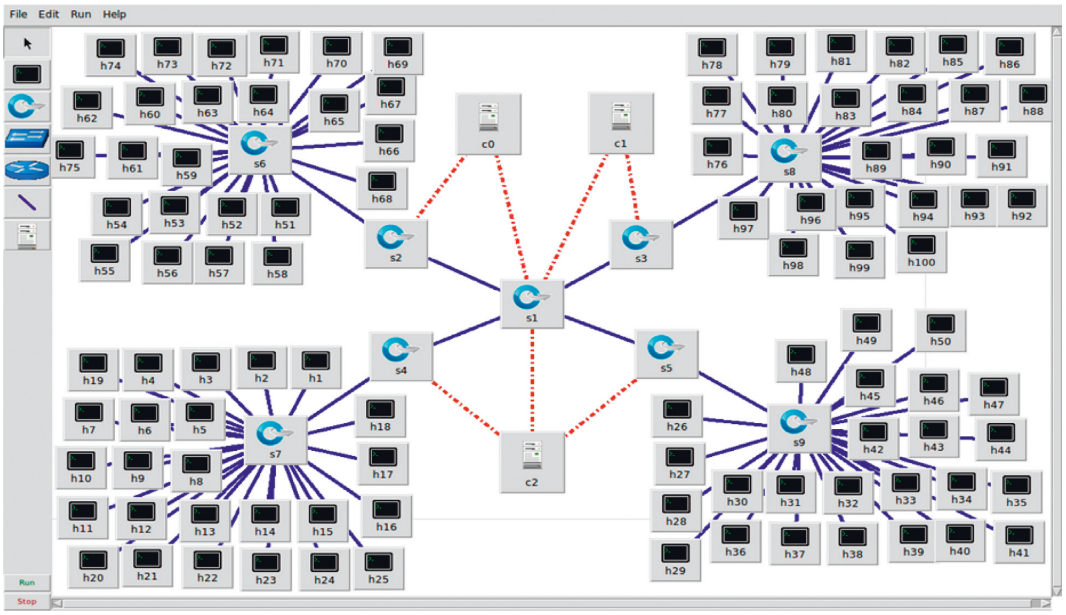


Figure 9: SDN testbed for detection of DDoS attack by using ASVM.

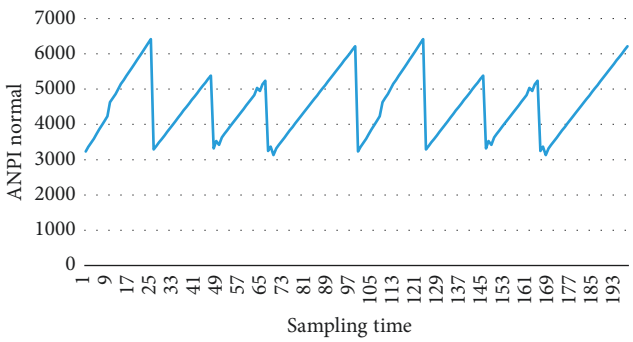


Figure 10: Feature of ANPI for normal traffics.

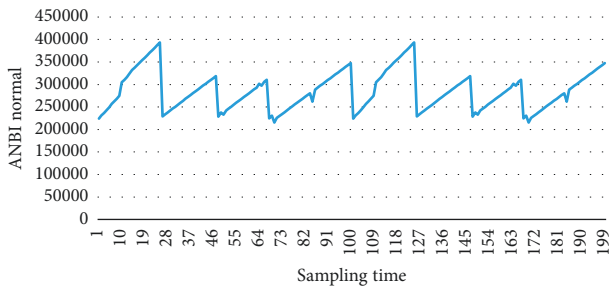


Figure 12: Feature of ANBI for normal traffics.

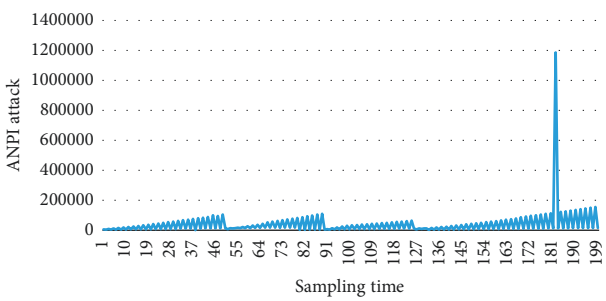


Figure 11: Feature of ANPI for attack traffics.

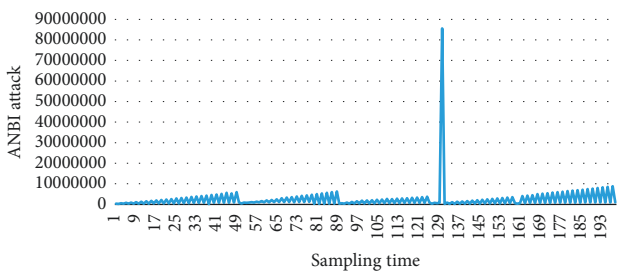


Figure 13: Feature of ANBI for attack traffics.

First, SDNtrafficDS is read and the Type field and the last fields is separated. The data is then split into Training DS and Testing DS using a cross-validation method in order to reduce an overfitting problem [34]. Next, the model is produced by ASVM using the Training DS. Linear kernel, OVS decision function, classification error "C" ($C > 0$), and the auto Gamma value are used in our ASVM. After the training process is done, the resulting model is used for classifying the Testing DS. The confusion matrix is used for the performance evaluation of the classification results. The classification report for three classes is generated. Lastly, the accuracy of our proposed classification result from the Training DS and the Testing DS is also generated.

11. Evaluation of Prediction Result

Training DS and testing DS are multidimensional data. We have solved our research's first problem of multiclass

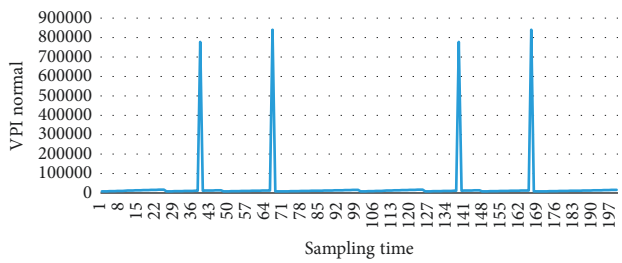


Figure 14: Feature of VPI for normal traffics.

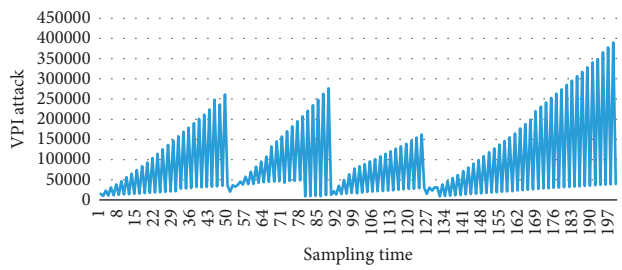


Figure 15: Feature of VPI for attack traffics.

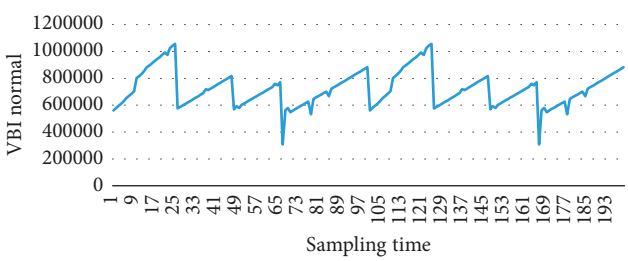


Figure 16: Feature of VBI for normal traffics.

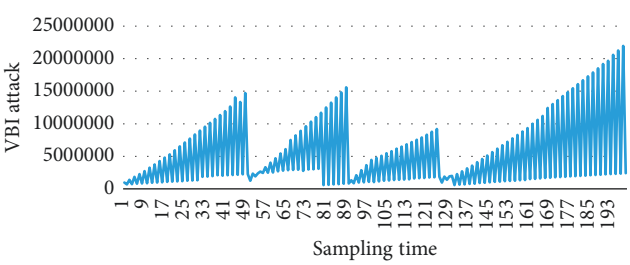


Figure 17: Feature of VBI for attack traffics.

problem extension. The second problem of the long training time and testing time of SVM algorithm has been solved by using the linear kernel with penalty parameter of the classification error term, 'C,' considering the value of "gamma" and "OVS" decision function shape. False alarm rate, detection rate, and accuracy are used for evaluating our detection result. False alarm rate is the error rate of our detection system that is the incorrect result on a normal behaviour. Thus, less false alarm rate is preferred. Detection rate is the correct rate for detecting the malicious traffics. The higher detection rate is the better system performance. Accuracy is the measurement of the system that correctly classifies both normal traffics and malicious

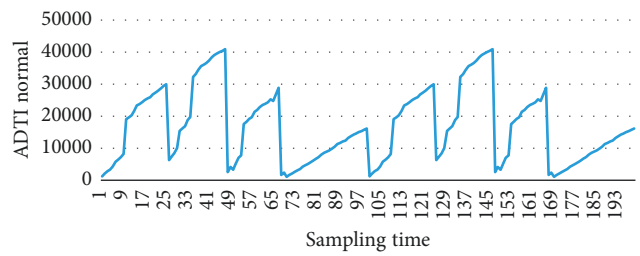


Figure 18: Feature of ADTI for normal traffics.

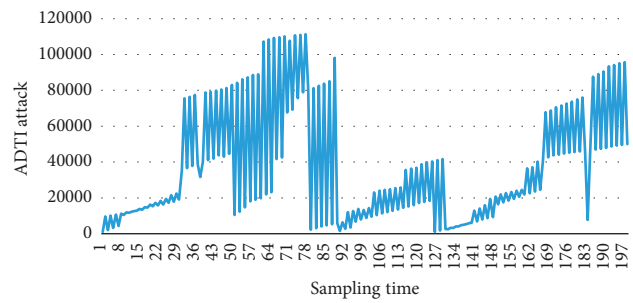


Figure 19: Feature of ADTI for attack traffics.

traffics. All three measures are shown in the following equations:

$$\text{false alarm rate} = \frac{\text{FP}}{\text{TP} + \text{FP}} * 100\%,$$
$$\text{detection rate} = \frac{\text{TP}}{\text{FN} + \text{TP}} * 100\%, \quad (10)$$
$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} * 100\%.$$

True positive (TP) is the amount of network traffics that are correctly detected attack or normal traffic and forwarded. True negative (TN) is the amount of network traffics that are correctly detected and dropped. False positive (FP) is the amount of network traffics that are incorrectly detected and forwarded. False negative (FN) is the amount of network traffics that are incorrectly detected and dropped. In this experiment, we have been trained and tested with the cross-validation method of splitting rate from 10% to 90% of SDNTrafficDS. The experimental result can be seen in Table 2.

According to the experimental results shown in Table 2, the average accuracy of the detection is 0.97, the average false alarm rate is 0.02, and the average detection rate is 0.97. The training time and testing time for each rate are approximately 50 seconds and 55 seconds, respectively.

12. Conclusion

In this paper, we proposed a way to detect two flooding-based DDoS attacks using the proposed advanced SVM method. Nowadays, most researches in the detection of the DDoS attack on SDN network used traditional networking dataset. In this work, a new dataset, SDNTrafficDS,

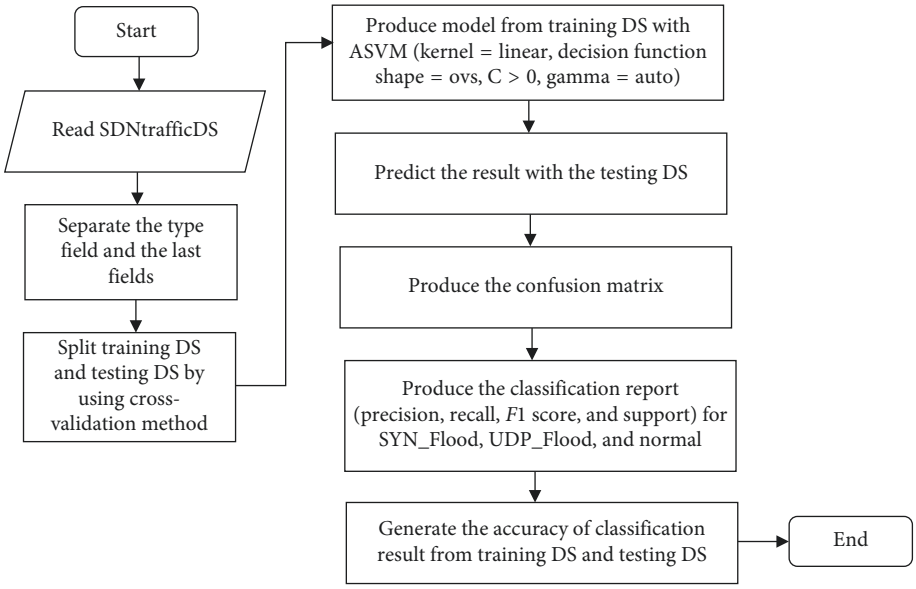


Figure 20: System flow of the proposed classification method.

Table 2: Evaluation of prediction performance of ASVM method.

Split rate	Training data (%)	Testing data (%)	False alarm rate	Detection rate	Accuracy
0.1	90	10	0	1.0	1.0
0.2	80	20	0.06	0.93	0.93
0.3	70	30	0.02	0.98	0.97
0.4	60	40	0.03	0.97	0.96
0.5	50	50	0.01	0.99	0.98
0.6	40	60	0.01	0.99	0.98
0.7	30	70	0.01	0.99	0.99
0.8	20	80	0.03	0.96	0.96
0.9	10	90	0.03	0.97	0.97

is generated and used. Our emulated testbed is conducted using Mininet. In our testbed, one hundred hosts, nine switches, and three controllers are used. The existing researches in the security of SDN network used a single controller in their network setting. In this work, on the other hand, three controllers are used. Although one controller has down because of the attack, another controller can still be used. We used one hundred scenarios for UDP flooding attack and another one hundred scenarios for SYN flooding attacks. Both malicious traffic data and normal traffic data are generated. The SDN traffics from the OpenFlow switches are collected. The volumetric and asymmetric features from the SDN traffics are collected and extracted to create the dataset. Cross-validation method is employed while training and testing the classification model. Linear kernel is used in our SVM algorithm. As a result, the training and testing time is reduced. The parameter of classification error (C), gamma value, and decision function shape (OVS) is considered. According to the experimental results, the overall accuracy of the proposed model is at 97%. Our future works include an online detection system for DDoS attack on SDN network. In addition, other attack planes of SDN layer must also be considered. Moreover, we would like to mitigate the DDoS attack using the lightweight method.

Data Availability

We have used our own dataset by using Mininet emulator. Our dataset is available at https://my.pcloud.com/publink/show?code=XZYm5P7ZXWd1JwSha2XTmPMtkfv2wzdXp5my.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors would like to express their gratitude to their scholarship program. This work was supported by the Higher Education Research Promotion and the Thailand's Education Hub for Southern Region of ASEAN Countries Project Office of the Higher Education Commission. Furthermore, the authors would like to thank all colleagues from the CNR Lab, Department of Computer Engineering at the Prince of Songkla University.

References

[1] M. Myint Oo, K. Sinchai, and K. Thossaporn, "The design of SDN based detection for distributed denial of service (DDoS)

attack," in *Proceedings of the 21st International Computer Science and Engineering Conference*, Antalya, Turkey, August 2017.

[2] T. Dang-Van and H. Truong-Thu, "A multi-criteria based software defined networking system Architecture for DDoS-attack mitigation," *REV Journal on Electronics and Communications*, vol. 6, no. 3-4, 2016.

[3] T. Evgeniou and M. Pontil, "Support vector machines: theory and applications," *Machine Learning and Its Applications: Advanced Lectures*, vol. 2049, pp. 249–257, 2001.

[4] S. Badotra and J. Singh, "Open daylight as a controller for software defined networking," *International Journal of Advanced Computer*, vol. 8, no. 5, 2017.

[5] S. Kolahi, K. Treseangrat, and B. Sarrafpour, "Analysis of UDP DDoS flood cyber-attack and defense mechanisms on Web Server with Linux Ubuntu 13," in *Proceedings of the 2015 International Conference on Communications, Signal Processing, and their Applications (ICCSPA'15)*, London, UK, June 2015.

[6] R. Bani-Hani and Z. Al-Ali, "SYN flooding attacks and countermeasures: a survey," in *Proceedings of ICICS*, Beijing, China, 2013.

[7] F. Gharvirian and A. Bohlooli, "Neural network based protection of software defined network controller against distributed denial of service attacks," *International Journal of Engineering*, vol. 30, no. 11, pp. 1714–1722, 2017.

[8] R. T. Kokila, S. Thamarai Selvi, and G. Kannan, "DDoS detection and analysis in SDN-based environment using support vector machine classifier," in *Proceedings of the 2014 Sixth International Conference on Advanced Computing (ICoAC)*, Chennai, India, December 2014.

[9] Y. Chi Wu, H. Tseng, W. Yang, and R. Hong Jan, "DDoS detection and traceback with decision tree and grey relational analysis," in *Proceedings of the 2009 Third International Conference on Multimedia and Ubiquitous Engineering*, Qingdao, China, June 2009.

[10] L. Linxia, V. C. M. Leung, and L. Chin-Feng, "Evolutionary algorithms in software defined networks: techniques, applications, and issues," *ZTE Communications*, vol. 15, no. 3, 2017.

[11] N. Anandshree Singh, K. Johnson Singh, and T. De, "Distributed denial of service attack detection using naive bayes classifier through info gain feature selection," in *Proceedings of the International Conference on Informatics and Analytics*, Pondicherry, India, August 2016.

[12] M. I. W. Pramana, Y. Purwanto, and F. Yosef Suratman, "DDoS detection using modified K-means clustering with chain initialization over landmark window," in *Proceedings of the 2015 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC)*, Bandung, Indonesia, August 2015.

[13] K. Benzekki, A. El Fergougui, and A. Elbelrhiti Elalaoui, "Software-defined networking (SDN): A survey," *Security and Communication Networks*, 2017.

[14] S. Kazuya, S. Kentaro, T. Nobuyuki et al., "A survey on OpenFlow technologies," *IEICE Transactions on Communications*, vol. E97.B, no. 2, pp. 375–386, 2014.

[15] N. Zakaria Bawany and J. A. Shamsi, "Application layer DDoS attack defense framework for Smart city using SDN," in *Third International Conference on Computer Science, Computer Engineering, and Social Media (CSCESM)*, Thessaloniki, Greece, May 2016.

[16] N. Dayal, P. Maity, S. Srivastava, and R. Khondoker, "Research trends in security and DDoS in SDN," *Security and Communication Networks*, vol. 9, no. 18, pp. 6368–6411, 2016.

[17] A. Akamai, "State of the internet/security," *SOTI*, vol. 4, no. 5, 2018.

[18] A. Akamai, *Memcached Reflection Attacks: A NEW era for DDoS*, Akamai Technologies, Cambridge, MA, USA, 2018.

[19] S. Acharya and N. Tiwari, "Survey of DDoS attacks based on TCP/IP protocol vulnerabilities," *IOSR Journal of Computer Engineering*, vol. 18, no. 3, pp. 68–76, 2016.

[20] M. Bogdanoski, A. Risteski, and T. Shuminoski, "TCP SYN flooding attack in wireless networks," in *Proceedings of the Conference: Innovations on Communication Theory, INCT*, Istanbul, Turkey, October 2012.

[21] S. H. Mujtiba and G. R. Beigh, "Impact of DDoS attack (UDP flooding) on queuing models," in *Proceedings of the 2013 4th International Conference on Computer and Communication Technology (ICCCT)*, Allahabad, India, September 2013.

[22] H. Harshita, "Detection and prevention of ICMP flood DDOS attack," *International Journal of New Technology and Research (IJNTR)*, vol. 3, no. 3, pp. 63–69, 2017.

[23] A. Verma and D. Kumar Xaxa, "A survey on HTTP flooding attack detection and mitigating methodologies," *International Journal of Innovations and Advancement in Computer Science*, vol. 5, no. 5, 2016.

[24] F. Yihunie, A. Eman, and A. Odeh, "Analysis of ping of death DoS and DDoS attacks," in *Proceedings of IEEE Long Island Systems*, Applications and Technology Conference (LISAT), Farmingdale, NY, USA, May 2018.

[25] S. Rajneet, "A study of DoS & DDoS-smurf attack and preventive measures," *International Journal of Computer Science and Information Technology Research*, vol. 2, no. 4, 2014.

[26] T. Lukaseder, G. Shreya, and K. Frank, "Mitigation of flooding and slow DDoS attacks in a software-defined network," in *Proceedings of Cryptography and Security*, Santa Barbara, CA, USA, August 2018.

[27] L. Tauber, "Introducing the normal distribution in a data analysis course: specific meaning contributed by the use of computers," in *Proceedings of the ICOTS 6 : the Sixth International Conference on Teaching Statistics*, Cape Town, South Africa, 2002.

[28] F. Tang, P. Tiňno, P. A. Gutiérrez, and H. Chen, "The benefits of modelling slack variables in SVMs," *Neural Computation*, vol. 27, no. 4, 2015.

[29] M.-F. Balcan, A. Blum, and S. Vempala, "On kernels, margins, and low-dimensional mappings," *Lecture Notes in Computer Science*, in *Proceedings of the 15th International Conference on Algorithmic Learning Theory*, Padova, Italy, October 2004.

[30] Y. Ahuja and S. K. Yadav, "Multiclass classification and support vector machine," *Global Journal of Computer Science and Technology Interdisciplinary*, vol. 12, no. 11, 2012.

[31] S. Asadollahi, B. Goswami, and A. M. Gonsai, "Implementation of SDN using OpenDayLight controller," in *Proceedings of the International Conference on Recent Trends in IT Innovations-Tecáfe*, vol. 52, no .2, India, April 2017.

[32] F. Keti and S. Askar, "Emulation of software defined networks using mininet in different simulation environments," in *Proceedings of the 6th International Conference on Intelligent Systems, Modeling, and Simulation*, Kuala Lumpur, February 2015.

[33] B. Pfaff, "Open vSwitch," 2014, http://www.openvswitch.org//support/slides/brkt.pdf.

[34] A. Moore, "Cross-validation for detecting and preventing overfitting," 2001.

Hindawi
Journal of Computer Networks and Communications
Volume 2019, Article ID 2182803, 10 pages
https://doi.org/10.1155/2019/2182803

Research Article

A Novel Hybrid Network Traffic Prediction Approach Based on Support Vector Machines

Wenbo Chen,[1] Zhihao Shang,[2] and Yanhua Chen[3]

[1]*School of Information Science and Engineering, Lanzhou University, Lanzhou 730000, China*
[2]*Department of Mathematics and Computer Science, Free University of Berlin, Berlin, Germany*
[3]*School of Information Engineering, Zhengzhou University, Zhengzhou 450000, China*

Correspondence should be addressed to Zhihao Shang; shangzh11@lzu.edu.cn

Received 28 September 2018; Revised 11 January 2019; Accepted 16 January 2019; Published 12 February 2019

Guest Editor: Saman S. Chaeikar

Network traffic prediction performs a main function in characterizing network community performance. An approach which could appropriately seize the salient characteristics of the network visitors could be very useful for network analysis and simulation. Network traffic prediction methods could be divided into two classes: one is the single models and the opposite is the hybrid fashions. The hybrid models integrate the merits of several single models and consequently can enhance the network traffic prediction accuracy. In this paper, a new hybrid network traffic prediction method (EPSVM) primarily based on Empirical Mode Decomposition (EMD), Particle Swarm Optimization (PSO), and Support Vector Machines (SVM) is presented. The EPSVM first utilizes EMD to eliminate the impact of noise signals. Then, SVM is applied to model training and fitting, and the parameters of SVM are optimized by PSO. The effectiveness of the presented method is examined by evaluating it with different methods, including basic SVM (BSVM), Empirical Mode Decomposition processed by SVM (ESVM), and SVM optimized by Particle Swarm Optimization (PSVM). Case studies have demonstrated that EPSVM performed better than the other three network traffic prediction models.

1. Introduction

It is generally known that network traffic prediction can provide a variety of practical information for Internet organizations, for example, about travelling, rental company, and smart search. Network traffic prediction is a procedure whereby a webmaster catches the network traffic and inspects it closely to discover what is going to happen in the follow-up and coming period on the network. It can assist each webmaster by establishing reasonable network planning and controlling the network traffic congestion effectively [1]. Precise network traffic prediction can thoroughly catch the notable attributes of the traffic, and thus it plays a vital role in network traffic analysis and simulation and offers assistance to customers to understand the network dynamics. So, in recent years, to enhance the network traffic prediction precision, researchers in China and abroad have proposed numerous network traffic prediction methods.

In general, network traffic prediction methods can be divided into two categories: one is the single models and the other is the combination, i.e., hybrid model which integrates the merits of several single models [2]. Dickinson [3] has demonstrated that the combined and hybrid models can get better forecasting result than that of individual models. Besides, the topology and geometry of network are always very complex, which often influence the network traffic prediction accuracy. Demonstration of network traffic complexity shows up in numerous circumstances, for instance, the long-area connections and self-resemblance were found in a statistical analysis of traffic estimations. The complexity indicated from the traffic estimations has prompted the development of network traffic prediction,

which suggests that a single model cannot yield satisfactory prediction result [4–6]. The main reason behind this is that network traffic displays numerous characteristics, such as trend, cycle time, self-resemblance, and long-area dependence. Network traffic prediction with a single model cannot capture all the characteristics mentioned above. But a combination model can not only capture the linear characteristics but also the nonlinear characteristics of the NTD (NTD). Therefore, the combination model is applied in this paper.

Over the past few decades, scientists over China and abroad have presented a lot of strategies to predict network traffic in diverse areas [7, 8]. Among them, some were more inclined to improve the existing models. For example, the literature [9] prolonged the notion of the broadly stated and used the fractional Brownian traffic model. Qing-Fang et al. [10] used a BIC-based totally neighboring factor choice approach to select the quantity of the nearest neighboring factors for the nearby Support Vector Machines. And, with the intention to obtain quicker convergence in the training of BiLinear Recurrent Neural Network (BLRNN), the literature [11] applied two procedures to the network. Other experts preferred a combination of the existing models. For example, the literature [1] developed a novel combined model to predict the network traffic in the National Taitung University and Shu-Te University. Chen et al. [12] evolved a new bendy neural tree structure which used Gene Programming, and the parameters are optimized through the Particle Swarm Optimization algorithm. The literature [13] presented a novel approach, which integrated wavelet transform, the grey theory, and the chaos theory; the numerical experiment demonstrated that the proposed model can get better prediction results. Recent papers about network traffic prediction methods can be seen in literatures [14, 15].

Despite the fact that the previous mentioned approaches can produce an adequately precise prediction result for various cases, they, in general, have focused on the precision evaluation of the approaches obtained without noting the internal characteristics of the network traffic data (NTD). In truth, NTD are normally influenced by means of risky factors, therefore inflicting noise indicators that can increase the difficulty of forecasting. So in this paper, the EMD is first applied to eliminate the noise signals before applying SVM to predict the network traffic. Besides, the parameters in SVM are optimized by PSO. Therefore, the presented method integrates the EMD, PSO, and SVM, hence its abbreviated name is EPSVM. In order to examine the effectiveness of EPSVM, we contrast it with three other approaches, namely, (1) the original NTD directly processed by SVM (the method is named as BSVM), (2) NTD processed by EMD and then using SVM to model the denoised data (the method is named as ESVM), and (3) the original NTD directly processed by the SVM, whose parameters are optimized by PSO (the method is named as PSVM). Besides, it is noteworthy that the NTD are gathered from the Network Center of Lanzhou University.

The rest of this paper is presented as follows. The theoretical background of EMD, PSO, and SVM models is specified in Section 2. In Section 3, the presented approach is introduced. Section 4 illustrates the experimental results. At last, Section 5 concludes this paper.

2. Theoretical Background of EMD, PSO, and SVM Models

In this subsection, the theories related to the proposed method (EPSVM) are introduced, and they are EMD, PSO, and SVM.

2.1. Empirical Mode Decomposition. Empirical Mode Decomposition (EMD) is a nonlinear signal processing method developed by Huang et al. [15]. It can decompose a signal into a sum of functions and intrinsic mode functions (IMFs). These IMFs must satisfy two conditions: (1) the number of extrema and the number of zero-crossings either are equal or differing at most by one; (2) the mean value of the envelope defined by the local maxima and the local minima is zero at all points. According to [16–18], any signal $y(t)$ can then be disintegrated:

(1) Identify all the local extrema, and then connect all the local maxima with a cubic spline line as the upper envelope.

(2) Repeat the procedure for the local minima to produce the lower envelope. The upper and lower envelopes should cover all the data between them.

(3) The mean of the upper and lower envelopes is designated as $n_1(t)$, and the difference between the signal $y(t)$ and $n_1(t)$ is the first component $p_1(t)$:

$$p_1(t) = y(t) - n_1(t). \tag{1}$$

Ideally, if $p_1(t)$ satisfies the definition of an IMF, then it is the first IMF.

(4) If $p_1(t)$ is not an IMF, $p_1(t)$ is treated as the original signal, and by repeating processes (1), (2), and (3), $p_{11}(t) = p_1(t) - n_{11}(t)$ is acquired. After repeating the sifting process up to k times, p_{1k} becomes an IMF, i.e.,

$$p_{1(k-1)} - n_{1k} = p_{1k}. \tag{2}$$

Then, it is designated as

$$c_1 = p_{1k}. \tag{3}$$

The first IMF component from the original data c_1 should contain the finest scale or the shortest period component of the signal.

(5) Separating $c_1(t)$ from the original signal $y(t)$, we could get $r_1(t) = y(t) - c_1(t)$. By repeating the above process several times, the result was $r_i(t) = r_{i-1}(t) - c_i(t)$, $i = 2, 3, \ldots, n$. Then, $c_i(t)$, $i = 1, 2, \ldots, n$, are the n IMFs that were obtained.

2.2. Particle Swarm Optimization. Particle Swarm Optimization (PSO) is one of the recent meta-heuristic technologies proposed by Kennedy and Eberhart [19] in view of the natural flocking and swarming behaviors of birds and insects. Consider an optimization problem of M variables. A swarm comprises of q particles flying in a M dimensional search space. Let x denote a particle's position and v denote the particle's flight velocity over a solution space. Each individual x in the swarm is scored utilizing a scoring function that obtains a fitness value representing how good it settles the issue. The best previous position of a particle is *pbest*. The index of the best particle among all particles in the swarm is *gbest*. Each particle records its own personal best position (*pbest*) and knows the best positions found by all particles in the swarm (*gbest*). Then, the best position of particle i could be calculated [20]:

$$v_i^{k+1} = w \cdot v_i^k + c_1 \cdot r_1 \cdot \left(pbest_i^k - x_i^k\right) + c_2 \cdot r_2 \cdot \left(gbest_i^k - x_i^k\right), \tag{4}$$

$$x_i^{k+1} = x_i^k + \alpha \cdot v_i^k, \tag{5}$$

where w is the inertia weight factor, r_1 and r_2 are two independent randomly distributed variables with the range of [0, 1], and c_1 and c_2 are two positive constants called acceleration coefficients.

2.3. Support Vector Machines. Support Vector Machine (SVM) [21] is a set of classification and regression techniques, designed to systematically optimize its structure based on the input training data. More details about SVM can be seen in the literatures [22–24].

Given the training data $(x_1, y_1), \ldots, (x_n, y_n) \in W \times \mathbb{R}$, where W denotes the space of the input patterns x_i (e.g. $W = \mathbb{R}^n$) and y_i is the associated output values of x_i. In ε-SVR, our goal is to produce a function $\overline{F}(x)$ based on the training data set to approximate the unknown function $F(x)$. By introducing different constraints for violating a "tube" constraint from above and from below, we arrive at the formulation stated in Vapnik's article [25] for ε-SVR:

$$\min_{w,b,\xi,\xi^*} \frac{1}{2} w^T w + c \sum_{i=1}^{n} (\xi_i + \xi_i^*),$$
$$\text{s.t.} \begin{cases} y_i - (< w, x_i > + b) \le \varepsilon + \xi_i, \\ (< w, x_i > + b) - y_i \le \varepsilon + \xi_i^*, \\ \xi_i, \xi_i^* \ge 0, \end{cases} \tag{6}$$

where n denotes the number of samples, whereas ξ_i and ξ_i^* are the allowed error "above" and "below" the training error subject to ε-insensitive tube $|y_i - (< w, x_i > + b)| \le \varepsilon$ and $w^T w$ is the regularization term. The empirically selected constant $C > 0$ determines the tradeoff between these two terms.

To preserve the sparse property of the solution, Vapnik used the ε-insensitive loss function $|\xi_\varepsilon|$ described by

$$|\xi_\varepsilon| := \begin{cases} 0, & \text{if } |\xi| < \varepsilon, \\ |\xi| - \varepsilon, & \text{otherwise.} \end{cases} \tag{7}$$

Instead of minimizing the observed training error, ε-SVR attempts to minimize the generalization error bound so as to achieve generalized performance, and this makes ε-SVR extremely robust to outliers. Finally, we get the following explicit form by introducing Lagrange multipliers, the Kernel trick, and employing the optimality constraints:

$$\overline{F}(x, \alpha_i, \alpha_i^*) = \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) \kappa(x, x_i) + b. \tag{8}$$

3. The Proposed Method

The proposed method (EPSVM) first uses EMD to eliminate the noise signal, then the data after EMD procedure are put into the SVM, and the parameters of SVM are optimized by PSO. So, in this subsection, the theory of SVM optimized by PSO is introduced in Section 3.1. And then, the specific prediction procedure of EPSVM is presented in Section 3.2.

3.1. SVM Optimized by PSO. The parameters of SVM have an extraordinary effect on the forecasting precision, and it is very important to optimize the two parameters in the forecasting procedure. So, PSO is utilized to optimize the parameters in SVM (which is named as PSVM). The detailed process of PSVM is depicted in Figure 1 which includes the following five steps:

(1) Initialization: the quantity of the population q is initialized, and the preliminary position and velocity of each particle are randomly allocated.

(2) Fitness assessment: for each particle, its fitness is assessed, and the fitness function is calculated as the subsequent:

$$\text{Fitness} = \frac{1}{N} \sum_{i=1}^{N} \left| \frac{x_i - \widehat{x}_i}{x_i} \right|, \tag{9}$$

where x_i and $\widehat{x}_i$ stand for the actual and forecast values, respectively.

(3) Update $pbest_i$ and $gbest_i$ according to the fitness function results.

(4) Update the velocity of each particle v_i according to Equation (4) and the position of each particle x_i using Equation (5).

(5) Termination: the velocity and position of the particle are updated until the stop conditions are met.

3.2. The Specific Process of the Proposed Method. To address the issues of noise signals caused by many uncertainties, EMD is first applied to remove the noise section of the NTD, which has many merits as discussed in Section 2.1. After the processing of the EMD, ESPSVM applied the PSVM described aforementioned on the processed NTD series to get

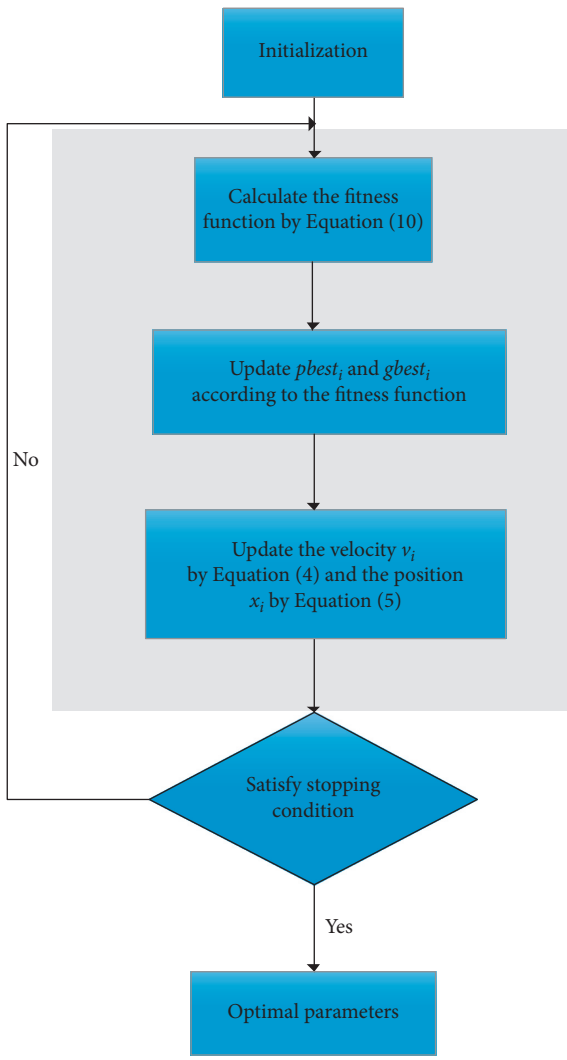


Figure 1: Flowchart of PSVM algorithm.

the final results. The detailed procedure of the EPSVM is depicted in Figure 2:

(1) Noise reduction: utilize EMD to remove the noise section of the original data.

(2) Put the data processed by the first step to PSVM model, and the final forecasting result can be obtained.

4. Experimental Results and Discussion

4.1. Criteria for Measuring Accuracy. In time series prediction, we always enquire what criterion may be used to correctly measure the accuracy of the anticipated outcomes. Performance evaluation of time series prediction is in fact tremendously depending on what sorts of the standards are chosen to measure the accuracy of predicted outcomes. In an effort to justify the affordable accuracy for a time series forecast, three famous criteria [26] are selected for measuring the prediction accuracy. The selected standards are expressed as follows:

$$\begin{aligned} \mathrm{RMSE} &= \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x}_i)}, \\ \mathrm{MAE} &= \frac{1}{N}\sum_{i=1}^{N}\left|x_i - \hat{x}_i\right|, \\ \mathrm{MAPE} &= \frac{1}{N}\sum_{i=1}^{N}\left|\frac{x_i - \hat{x}_i}{x_i}\right| \times 100\%, \end{aligned} \tag{10}$$

where N is the number of periods in forecasting and x_i and $\hat{x}_i$are the actual value and forecasted value.

4.2. Network Traffic Data (NTD). The presented EPSVM approach is evaluated by the real NTD in the Network Center of Lanzhou University (LZU). These data are gathered every 5 minutes, so each hour has 12 NTD, and one day amounts to 288 ($12 * 24$) NTD. In addition, the NTD on workdays and nonworkdays (7 days) are all applied to examine the effectiveness and feasibility of the proposed EPSVM method. Therefore, the total NTD used in this paper is 2016 ($288 * 7$). To guess the network traffic fluctuations instantly and at the same time expedite website tracking, two types NTD, namely, inflow and outflow NTD are applied. As a result, the prediction process contains two procedures: one being the inflow NTD prediction and the other the outflow NTD prediction. Figures 3 and 4 show the 2016 NTD which contains inflow NTD and outflow NTD, respectively.

From Figures 3 and 4, it can be concluded that the NTD used in this paper were divided into seven groups. Each group has 288 NTD, which means that each group represents one day. In the 288 data, the data from 12 p.m. to 7 a.m. (15 hours) are used for model training and fitting, and the trained as well as the fitted model is adopted to predict the NTD from 8 p.m. to 9 p.m.

4.3. Prediction Results. As discussed above, the EPSVM approach initially implemented the Empirical Mode Decomposition to put-off the noise interference from the authentic data. After the noise removal process of the authentic NTD, the records are named as denoised NTD. Figure 5 shows the noise putting-off technique of the inflow data and outflow data. It is worth noting that all of the NTD used was processed by EMD so as to observe its effect.

Through comparison of the authentic NTD with the denoised information from Figures 5 and 6, it could be seen that the denoised statistics becomes a little smoother. So, instead of the usage of the authentic series, the proposed approach EPSVM used the denoised information to model training and fitting. After acquiring the denoised data, EPSVM used the SVM version optimized with the PSO to predict further. Here, each institution of records became normalized, and every normalized record group is divided into training sets and testing units, where the training sets had been the NTD from 12 p.m. to 7 a.m. (15 hours) every day, and the testing units were the NTD from 8 p.m. to 9 p.m. The EPSVM obtained the final seven days prediction

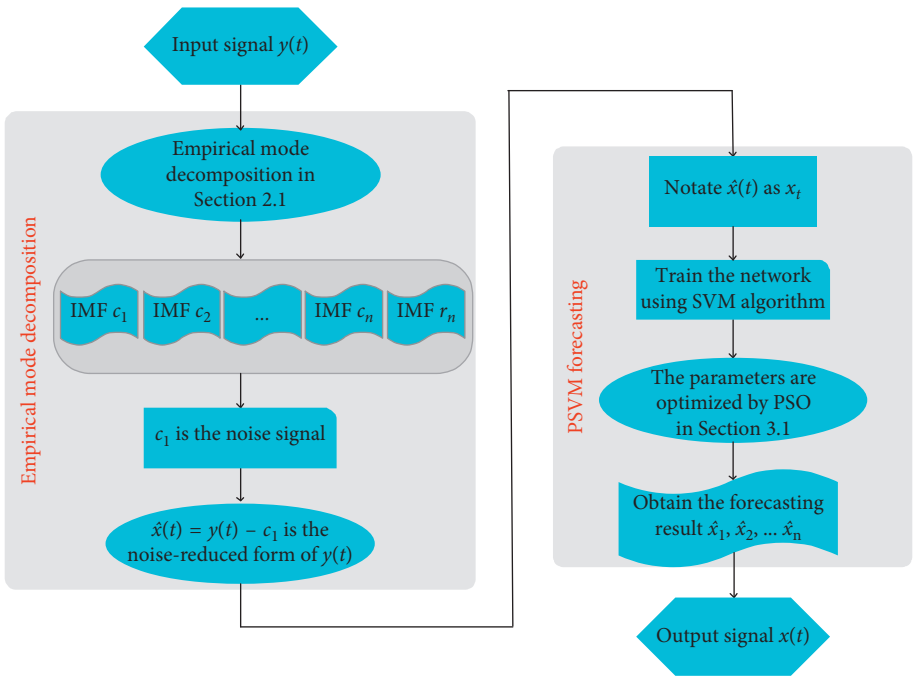


Figure 2: Flowchart of the proposed EPSVM algorithm.

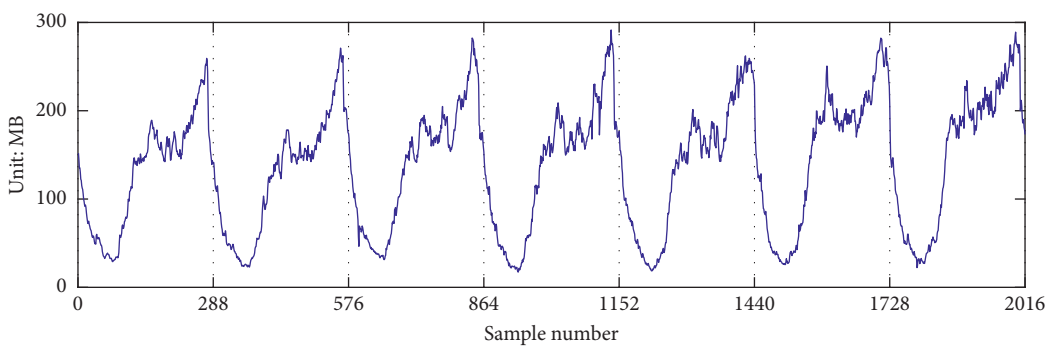


Figure 3: The inflow network traffic data.

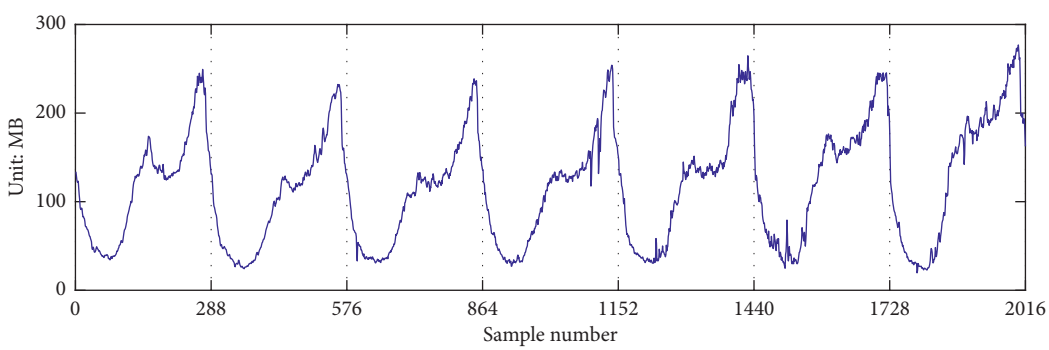


Figure 4: The outflow network traffic data.

results by predicting the 24 values of one day and forecasting them for the other six days in the week.

As for the other three methods, their common characteristics are that they all used SVM to model training. The difference is that the ESVM simulated the denoised data, BSVM and PSVM simulated the original data directly, but the parameters of PSVM are optimized by PSO. Figures 6 and 7 show the final inflow NTD and outflow NTD

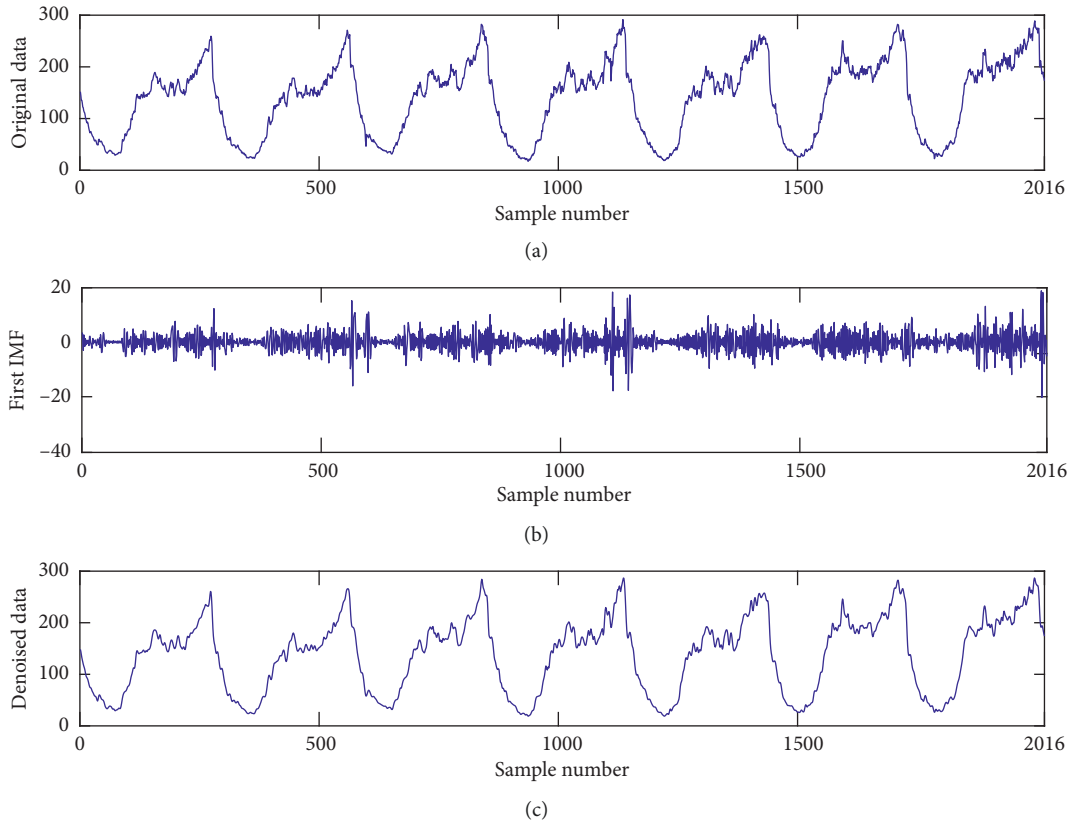


Figure 5: The noise eliminating process of inflow data.

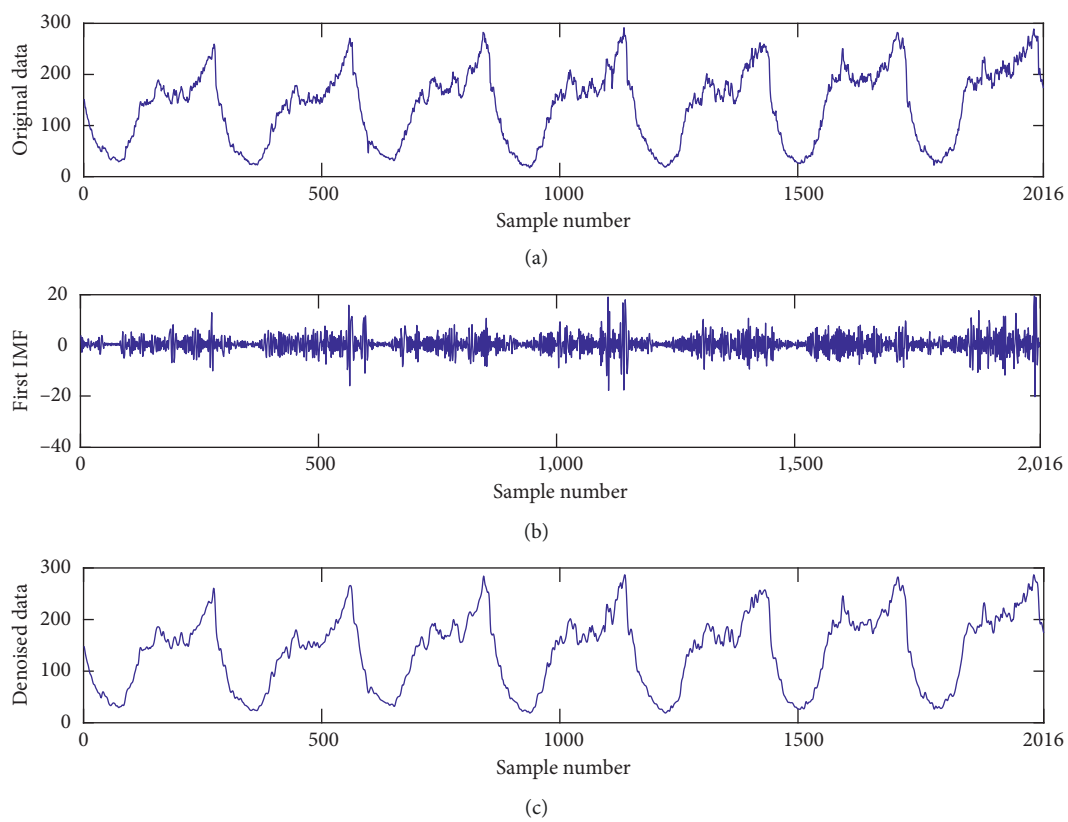


Figure 6: The noise eliminating process of outflow data.

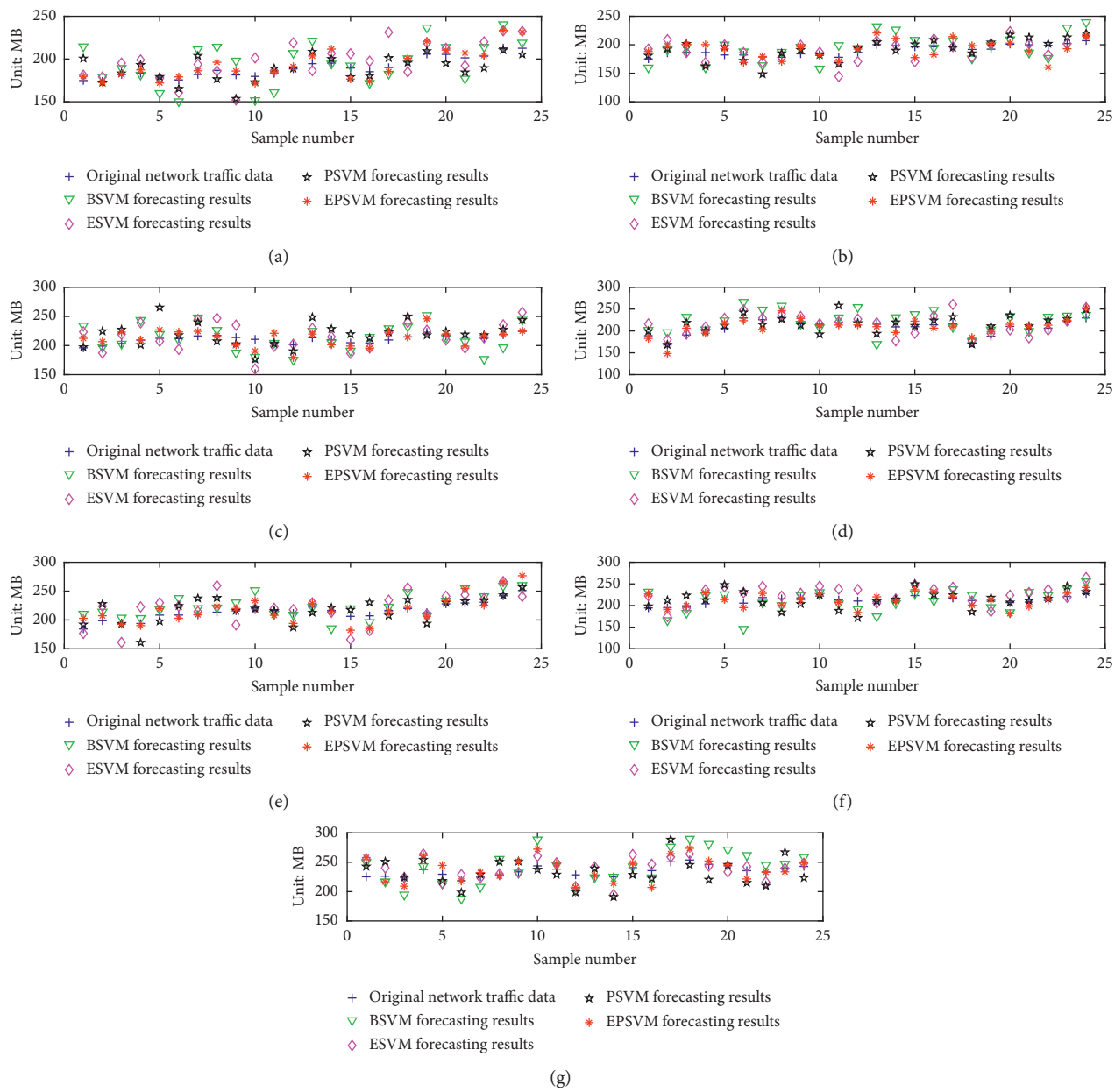


Figure 7: Forecasting result of inflow NTD at LZU network center: (a) Monday, (b) Tuesday, (c) Wednesday, (d) Thursday, (e) Friday, (f) Saturday, and (g) Sunday.

prediction results for each day by these four approaches, respectively.

Figures 7 and 8 just roughly show a contrast between the predicted results of the four methods and the original NTD to confirm that the EPSVM can perform better than the other three approaches. On the basis of the three evaluation metrics (RMSE, MAE, and MAPE) calculated in Section 4.1, the three criteria for measuring the accuracy of the four methods were computed in this section and are recorded in Tables 1 and 2.

Tables 1 and 2 show the three criteria results of the four approaches (BSVM, PSVM, ESVM, and EPSVM) on each day and the average values of the three criteria. Because of that, the results of the three metrics of the four forecasting methods in Table 1 are not the same as that in Table 2; we discuss the two tables separately. From Table 1, the subsequent outcomes occurred:

A comparison between BSVM and PSVM: Table 1 shows that if the three evaluation metrics of the seven days are all taken into consideration, PSVM had expected lower values than BSVM apart from Wednesday, Friday, and Saturday. There are only three days in the week; BSVM has lower values than PSVM; generally speaking, if we just compare the average values, it can be seen that PSVM performs better than BSVM.

A comparison between PSVM and ESVM: Table 1 shows that if the three metrics of each day are taken into consideration, the value of ESVM is lower values than PSVM for

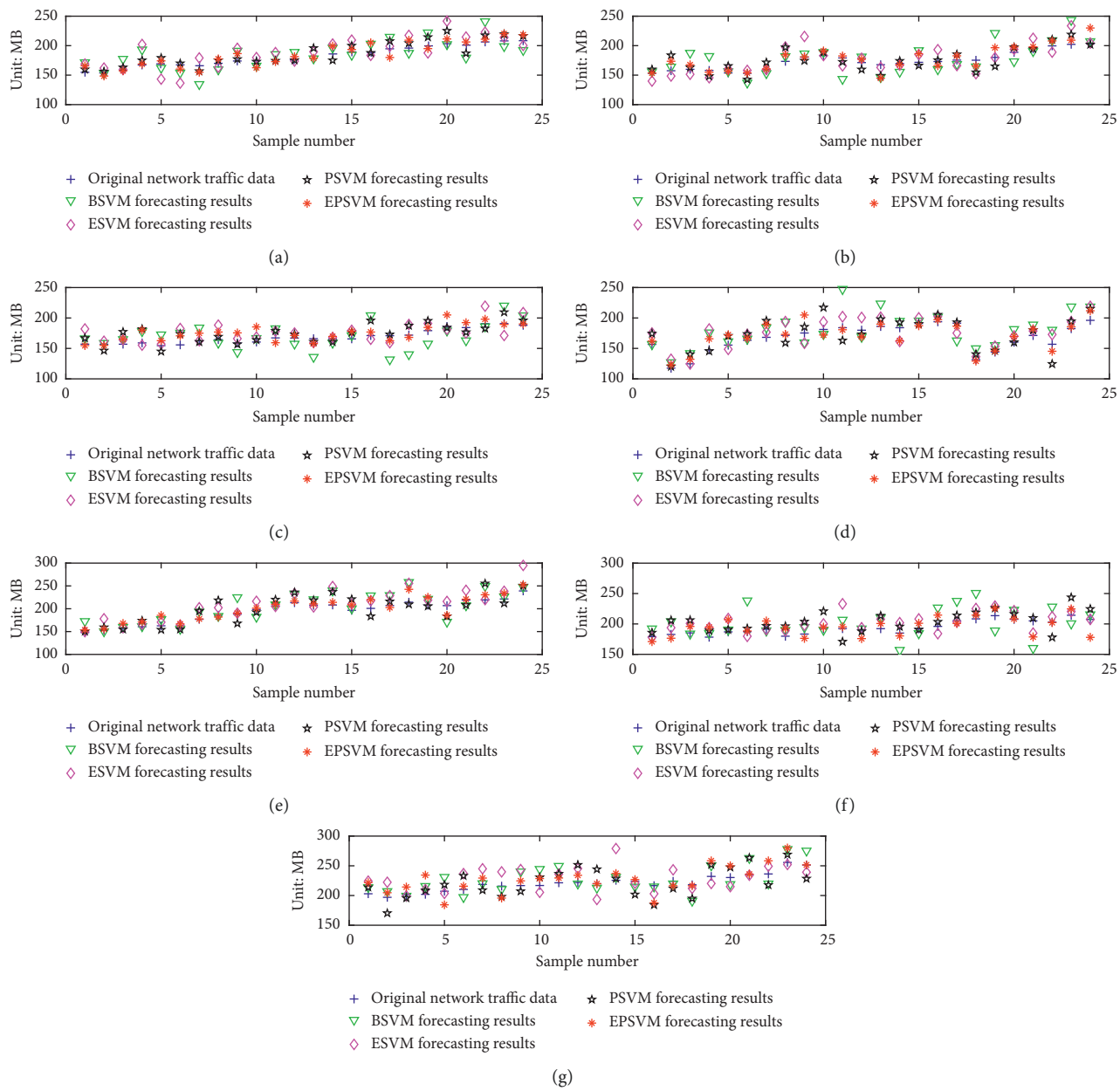


Figure 8: Forecasting result of outflow NTD at LZU network center: (a) Monday, (b) Tuesday, (c) Wednesday, (d) Thursday, (e) Friday, (f) Saturday, and (g) Sunday.

Table 1: The result of evaluation metrics of the four forecasting methods (inflow NTD).

Date	RMSE				MAE				MAPE			
	BSVM	PSVM	ESVM	EPSVM	BSVM	PSVM	ESVM	EPSVM	BSVM	PSVM	ESVM	EPSVM
Monday	20.082	17.033	11.994	9.262	16.822	14.226	9.298	7.369	0.0893	0.0747	0.0495	0.0379
Tuesday	18.086	14.846	12.202	13.892	15.834	11.772	10.274	11.649	0.0828	0.0625	0.0543	0.0608
Wednesday	20.668	20.784	20.555	12.335	16.825	17.086	16.185	10.067	0.0797	0.0808	0.0766	0.0482
Thursday	23.445	16.836	16.226	11.427	19.741	12.806	12.776	9.878	0.0948	0.0615	0.0609	0.0481
Friday	16.729	21.573	15.683	13.456	13.962	17.455	11.897	10.549	0.0654	0.0825	0.0571	0.0487
Saturday	21.784	22.186	18.898	13.915	18.422	19.146	14.877	10.977	0.0882	0.0904	0.0704	0.0524
Sunday	22.104	14.935	19.864	15.537	18.288	12.056	17.359	12.459	0.0776	0.0517	0.0741	0.0530
Whole week	20.414	18.313	16.489	12.832	17.128	14.935	13.238	10.421	0.0825	0.0720	0.0633	0.0498

Table 2: The result of the three evaluation metrics of the four forecasting methods (outflow NTD).

Date	RMSE				MAE				MAPE			
	BSVM	PSVM	ESVM	EPSVM	BSVM	PSVM	ESVM	EPSVM	BSVM	PSVM	ESVM	EPSVM
Monday	17.027	18.464	10.860	9.711	14.082	15.334	9.203	8.444	0.0786	0.0857	0.0498	0.0464
Tuesday	18.715	14.850	12.372	10.616	14.817	11.462	10.332	8.160	0.0848	0.0651	0.0604	0.0461
Wednesday	19.425	14.320	12.256	11.123	16.482	11.722	10.195	9.224	0.0973	0.0688	0.0608	0.0556
Thursday	22.645	15.693	15.471	11.918	18.160	13.933	12.275	9.330	0.1080	0.0831	0.0735	0.0560
Friday	21.152	20.504	17.132	11.974	16.312	15.764	14.322	9.333	0.0828	0.0783	0.0727	0.0480
Saturday	22.702	14.896	15.846	12.781	17.823	12.170	13.453	10.307	0.0907	0.0630	0.0693	0.0531
Sunday	16.793	19.053	20.077	16.235	14.011	16.379	17.356	13.229	0.0629	0.0749	0.0785	0.0604
Whole week	19.780	16.826	14.859	12.051	15.955	13.823	12.448	9.718	0.0864	0.0741	0.0664	0.0522

almost every day of the week apart from Sunday. And, in comparison with the average value of one week, it can be found that ESVM also had lower values than PSVM.

To sum up, ESVM performs better than PSVM, and PSVM was better than BSVM for most days of the week. If the advantages of PSVM and ESVM are assembled, the result should be superb. So, the proposed approach EPSVM which combined PSVM and ESVM could get better forecasting results. Table 1 shows that, if we compare the three metrics of each day, EPSVM had expected lower values than the alternative three methods for most days, apart from the fact that three evaluation metrics performances of EPSVM are higher than those of PSVM on Sunday and higher than those of EVM on Tuesday. In addition, if we compare the average values of EPSVM with other three alternative methods, EPSVM also has decrease values than the other three techniques.

From Table 2, the subsequent outcomes occurred:

A comparison between BSVM and PSVM: by observing Table 2, we see that if the three metrics of each day are all taken into consideration, PSVM had expected lower values than BSVM apart from Monday and Sunday. There are only two days in the week; BSVM has lower values than PSVM; generally speaking, if we just compare the average values, it can be seen that PSVM performs better than BSVM.

A comparison between PSVM and ESVM: by observing Table 2, we can see that if the three metrics of each day are considered, ESVM had expected lower values than PSVM apart from Saturday and Sunday. There are only two days in the week; PSVM has lower values than ESVM; generally speaking, if we just compare the average values, it can be seen that ESVM performs better than PSVM.

All in all, from the above statements deduced from Tables 1 and 2, the following conclusion can be reached: the proposed method EPSVM obviously performs better than the other three methods, and these four methods all have an acceptable performance for each day of the week.

5. Conclusions

Network traffic prediction offers useful data for website administrators to customize the records that are hosted on Internet servers with a view to reach a bigger target market. In an effort to decorate the functionality of real-time network visitor's analysis, it is very vital to expand a rather correct network visitor's prediction technique to help the webmaster control the bandwidth allocation effectively. In view of this, an artificial intelligence-based hybrid method EPSVM is presented in this article. EPSVM first uses EMD to process the original NTD so as to remove the noise part of the NTD. Then, it employs SVM to model the denoised network traffic series. Here, the parameters of SVM are optimized by PSO. Experiments with the NTD from LZU network center obviously verify that EPSVM significantly can enhance network traffic prediction accuracy. As part of real-time and reliable analysis of smart grids, EPSVM will help the webmaster better monitor the websites or, in other words, help network engineers optimize their websites, maximize online marketing, track user behavior, and push ads to users.

Data Availability

The data used to support the findings of this study were supplied by Lanzhou University under license and so cannot be made freely available. Requests for access to these data should be made to the corresponding author.

Conflicts of Interest

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

Acknowledgments

The authors would like to thank the Natural Science Foundation of China (61672469, 61472370, and 61822701), the Key Science and Technology Foundation of Gansu Province (1102FKDA010), and the Science and Technology Support Program of Gansu Province (1104GKCA037) for supporting this research.

References

[1] B. R. Chang and H. F. Tsai, "Improving network traffic analysis by foreseeing data-packet-flow with hybrid fuzzy-based model prediction," *Expert Systems with Applications*, vol. 36, no. 3, pp. 6960–6965, 2009.

[2] J. M. Bates and C. W. J. Granger, "The combination of forecasts," *Journal of the Operational Research Society*, vol. 20, no. 4, pp. 451–468, 1969.

[3] J. P. Dickinson, "Some comments on the combination of forecasts," *Operational Research Quarterly*, vol. 26, no. 1, pp. 205–210, 1975.

[4] G. Orosz, B. Krauskopf, and R. E. Wilson, "Bifurcations and multiple traffic jams in a car-following model with reaction-time delay," *Physica D: Nonlinear Phenomena*, vol. 211, no. 3-4, pp. 277–293, 2005.

[5] G. Orosz, R. E. Wilson, and B. Krauskopf, "Global bifurcation investigation of an optimal velocity traffic model with driver reaction time," *Physical Review E*, vol. 70, no. 2, article 026207, 2004.

[6] I. Gasser, G. Sirito, and B. Werner, "Bifurcation analysis of a class of "car following" traffic models," *Physica D: Nonlinear Phenomena*, vol. 197, no. 3-4, pp. 222–241, 2004.

[7] X. Wang, C. Zhang, and S. Zhang, "Modified elman neural network and its application to network traffic prediction," in *Proceedings of 2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems*, vol. 2, pp. 629–633, IEEE, Hangzhou, China, October-November 2012.

[8] W. Pan, Z. Shun-Yi, and C. Xue-Jiao, "SFARIMA: a new network traffic prediction algorithm," in *Proceedings of 2009 First International Conference on Information Science and Engineering*, pp. 1859–1863, IEEE, Nanjing, China, 2009.

[9] F. H. T. Vieira, G. R. Bianchi, and L. L. Lee, "A network traffic prediction approach based on multifractal modeling," *Journal of High Speed Networks*, vol. 17, no. 2, pp. 83–96, 2010.

[10] M. Qing-Fang, C. Yue-Hui, and P. Yu-Hua, "Small-time scale network traffic prediction based on a local support vector machine regression model," *Chinese Physics B*, vol. 18, no. 6, pp. 2194–2199, 2009.

[11] D. C. Park, "Structure optimization of BiLinear recurrent neural networks and its application to ethernet network traffic prediction," *Information Sciences*, vol. 237, pp. 18–28, 2013.

[12] Y. Chen, B. Yang, and Q. Meng, "Small-time scale network traffic prediction based on flexible neural tree," *Applied Soft Computing*, vol. 12, no. 1, pp. 274–279, 2012.

[13] S. Han-Lin, J. Yue-Hui, C. Yi-Dong, and C. Shi-Duan, "Network traffic prediction by a wavelet-based combined model," *Chinese Physics B*, vol. 18, no. 11, pp. 4760–4768, 2009.

[14] L. Nie, X. Wang, L. Wan et al., "Network traffic prediction based on deep belief network and spatiotemporal compressive sensing in wireless mesh backbone networks," *Wireless Communications and Mobile Computing*, vol. 2018, Article ID 1260860, 10 pages, 2018.

[15] N. E. Huang, Z. Shen, S. R. Long et al., "The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis," *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, vol. 454, no. 1971, pp. 903–995, 1998.

[16] L. Qian, G. Xu, W. Tian, and J. Wang, "A novel hybrid EMD-based drift denoising method for a dynamically tuned gyroscope (DTG)," *Measurement*, vol. 42, no. 6, pp. 927–932, 2009.

[17] C. Junsheng, Y. Dejie, and Y. Yu, "A fault diagnosis approach for roller bearings based on EMD method and AR model," *Mechanical Systems and Signal Processing*, vol. 20, no. 2, pp. 350–362, 2006.

[18] L. Yu, S. Wang, and K. K. Lai, "Forecasting crude oil price with an EMD-based neural network ensemble learning paradigm," *Energy Economics*, vol. 30, no. 5, pp. 2623–2635, 2008.

[19] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, vol. 1, pp. 39–43, Nagoya, Japan, October 1995.

[20] M. S. Kıran, E. Özceylan, M. Gündüz, and T. Paksoy, "A novel hybrid approach based on particle swarm optimization and ant colony algorithm to forecast energy demand of Turkey," *Energy Conversion and Management*, vol. 53, no. 1, pp. 75–83, 2012.

[21] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the Fifth Annual Workshop On Computational Learning Theory*, pp. 144–152, ACM, Pittsburgh, PA, USA, July 1992.

[22] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, no. 3, pp. 199–222, 2004.

[23] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[24] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*, Cambridge University Press, Cambridge, UK, 2000.

[25] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer, Berlin, Germany, 1995.

[26] F. Diebold, *Elements of Forecasting*, Cengage Learning, Boston, MA, USA, 2006.

Hindawi
Journal of Computer Networks and Communications
Volume 2019, Article ID 4612474, 9 pages
https://doi.org/10.1155/2019/4612474

Research Article

Malicious Domain Names Detection Algorithm Based on *N*-Gram

Hong Zhao,[1] **Zhaobin Chang**,[1] **Guangbin Bao**,[1] **and Xiangyan Zeng**[2]

[1]*School of Computer and Communication, Lanzhou University of Technology, Lanzhou 730050, China*
[2]*Department of Mathematics and Computer Science, Fort Valley State University, Fort Valley, GA 31030, USA*

Correspondence should be addressed to Hong Zhao; 594286500@qq.com

Received 21 November 2018; Accepted 15 January 2019; Published 3 February 2019

Guest Editor: Saman S. Chaeikar

Malicious domain name attacks have become a serious issue for Internet security. In this study, a malicious domain names detection algorithm based on *N*-Gram is proposed. The top 100,000 domain names in Alexa 2013 are used in the *N*-Gram method. Each domain name excluding the top-level domain is segmented into substrings according to its domain level with the lengths of 3, 4, 5, 6, and 7. The substring set of the 100,000 domain names is established, and the weight value of a substring is calculated according to its occurrence number in the substring set. To detect a malicious attack, the domain name is also segmented by the *N*-Gram method and its reputation value is calculated based on the weight values of its substrings. Finally, the judgment of whether the domain name is malicious is made by thresholding. In the experiments on Alexa 2017 and Malware domain list, the proposed detection algorithm yielded an accuracy rate of 94.04%, a false negative rate of 7.42%, and a false positive rate of 6.14%. The time complexity is lower than other popular malicious domain names detection algorithms.

1. Introduction

While rapid development of Internet has changed our lives positively, different types of malicious cyberattacks have been increasing simultaneously. According to the 36th issue of the 2018 "Network information security and dynamic weekly report" released by CNCERT, Chinese servers were attacked 178,156 times in 2018 by 3,724 malicious domain names such as Conficker, Trojan, and Srizbis, an year-over-year increase of 55.8% [1].

DNS (domain name system), one of the basic services for realizing the conversion between network domain name and IP addresses in the Internet [2], has been widely used in e-commerce, instant messaging, and network media. Almost all Internet applications are needed to use DNS to resolve domain names and achieve resource location [3, 4].

In order to achieve malicious purpose, attackers implant malicious programs through the vulnerabilities of system or service to infect the host, and the infected host is controlled by attackers remotely [5]. The infected host will issue resolution requests, using a large number of nonexistent domain names randomly generated by the DGA (domain generation algorithm) or domain flux [6] in a short time. These resolution requests of malicious domain name are forwarded multiple times among DNS servers and are discarded eventually due to the failure of domain name resolution. However, the record of the failure of domain name resolution is also forwarded multiple times, then returned to the infected host that initiates the domain name resolution request. A large number of resolution requests and resolution failure records of the malicious domain name are forwarded multiple times among the DNS servers, which increases the usage of network bandwidth and brings a heavy payload on the DNS servers. Moreover, it will affect the execution of normal domain name resolution tasks seriously as well. If the malicious domain name is not detected in an accurate and timely manner, the DNS servers may be down due to malicious domain name attacks, all Internet services relying on DNS servers will stop, and the results will be catastrophic. Therefore, accurate and timely detection of malicious domain name attacks has the significant impact on Internet security.

The remaining of this study is organized as follows. A number of related works are reviewed in Section 2. The proposed approach, system architecture, and progress to detect malicious domain names are presented in Section 3.

The experimental results and performance evaluation of the study are presented in Section 4, and the conclusion is presented in Section 5.

2. Related Works

From the perspective of domain name structure features and lexical composition, there are two main types of malicious domain names detection methods in the literature: domain name model [7–11] and domain name semantic [12–15] detection.

2.1. Domain Name Model Detection. There are many differences between the normal domain names and the malicious domain names in terms of behaviors and structures. Therefore, the legitimacy of domain names can be determined by analyzing the behavior and structure of domain names. For example, Truong et al. [16] used the DNS traffic characteristics to detect the DNS query flow of abnormal DNS servers. Zang et al. [17] proposed a malicious domain names detection algorithm based on AGD (algorithmically generated domain) by using cluster correlation that identifies the domain names generated by a domain generation algorithm or its variants. Features such as TTL, the distribution of IP addresses, Whois features, and historical information from the domain names in each cluster were extracted, and the support vector machine (SVM) algorithm was used to identify the malicious domain names. Sharifnya and Abadi [18] proposed a DGA-based botnet detection algorithm by clustering the hosts of the DNS request query and generating the potential candidate set to be tested. Their algorithm achieves malicious domain names detection by calculating the probability of the threatened candidate hosts in the candidate set to be tested. Kwon et al. [19] proposed a botnet detection algorithm based on DNS traffic features using PSD (power spectral density) testing technology, which detects malicious domain names by analyzing malicious behavior within large volumes of DNS traffic. Zhang et al. [20] proposed a botnet detection algorithm that combined the domain names' request behaviors with construction characteristics and conducted malicious domain names detection through SVM classification. Vinayakumar et al. [21] proposed a method that used big data computation platforms on a distributed cluster and deep learning algorithm to detect fraud and malicious activities where attackers used combinational method to avoid blacklist detection.

2.2. Domain Name Semantic Detection. The method of domain name semantic detection includes detection based on character matching and on content analysis. For example, Yadav et al. [22] proposed a malicious domain names detection algorithm by using linguistic features that measures information entropy of bigrams in all domains and statistical measurements such as Kullback–Leibler divergence, Jaccard index, and Levenshtein edit distance for identifying malicious domain names. Huang et al. [23] analyzed the differences between normal domain names and malicious domain names in character constitution. The statistical characteristics, resolution features, and similarity characteristics of domain names were extracted, and malicious domain names detection was achieved using a machine learning algorithm in the character and resolution features space. Zhang et al. [24] proposed a lightweight domain names detection algorithm based on morpheme features and natural language processing (NLP), which analyzed the domain name features such as the root, affix, Chinese spelling, and special noun abbreviation and used the C4.5 algorithm to construct a decision tree with recursive thinking. Zhang et al. [25] proposed a malicious domain names detection algorithm that analyzed the domain name features of character composition and the lexical hierarchical structure which included domain name length, double letters, and character frequency to distinguish malicious domain names. Zhao [26] proposed a high efficiency pattern matching detection algorithm for intrusion detection. The algorithm reduces computational time by constructing a hash table for attack pattern strings.

Each of the above malicious domain names detection methods has its own advantages. The domain name model detection methods have high detection accuracy rate and wide application range. However, this kind of detection method has a long data collection period, and it is difficult to obtain a large amount of resolution data from both the local domain name server and the root domain name server, thus resulting in high detection time overhead. Although the detection methods of domain name semantic have the advantage of low detection time overhead, this kind of detection method is based on domain name blacklist to design detection features and cannot effectively detect newly generated domain names.

To address the problems such as low detection accuracy rate and high detection overhead, a new method of malicious domain names detection based on N-Gram is proposed by combining domain name model and semantic features. In addition, unlike current methods that analyze the lexical composition and structure of the whole domain names, the new method divides the whole domain names into multiple domain name substrings and deeply analyzes the features of domain name substrings in terms of lexical composition and structure. In this algorithm, the domain names which are accessed with high frequency are chosen as the normal domain names sample, and the N-Gram method is applied to segment each domain name in the domain name whitelist sample to obtain the substrings on which domain name model and semantic features depend. Then, a test domain name is also segmented by the N-Gram method, and its substrings are compared with the domain name substrings in domain name whitelist substring set to determine whether it is a malicious domain name.

3. Proposed Approach

In the following sections, we provide more details for the components of the proposed algorithm.

3.1. System Overview. Figure 1 gives a flowchart of our proposed detection algorithm. Malicious domain names detection algorithm based on *N*-Gram consists of two main sections. (1) Domain name whitelist substring set construction. (2) Malicious domain names detection. There are two steps to construct the domain name whitelist substring set, namely, obtaining the substring statistics from the normal domain names segmented by the *N*-Gram method and calculating the substrings weight values. The domain names with high frequency of access excluding the top-level domain are segmented into multiple substrings according to its domain name level with the lengths of 3, 4, 5, 6, and 7 by using the *N*-Gram method. Then, substring statistics is calculated according to each domain name substring repetition occurrence number. The substring weight value is calculated according to the domain name substring occurrence frequency in the substring set.

In the process of malicious domain names detection, a test domain name is segmented also by the *N*-Gram method, and its reputation value is calculated according to the weight values of its substrings. Then, the malicious domain name is determined according to the reputation value.

3.2. Domain Name Whitelist Substring Set Construction. Alexa ranking is a service that Amazon provides to the public to evaluate the popularity of domain names [27]. Through the statistics and analysis of each domain name in the number of access, links, and other aspects, the domain name evaluation and ranking are produced according to the analysis results. Therefore, if a domain name ranks relatively high in the Alexa, it is more likely to be secure and normal.

Through the observation of a large number of normal domain names in Alexa 2013 [28], it is found that the domain name has the features of hierarchical structure on its composition form. Top-level domain is the domain name substring of the end of a domain name, including the country top-level domains (such as cn, jp, and gb) and international organization top-level domains (such as com, net, and org). Moreover, the total number of top-level domains is small, and there are unified naming rules in terms of its lexical structure and lexical composition [29]; namely, top-level domains are generally named on the basis of the name of country or region. For example, for a given domain name lut.edu.cn, cn is China's top-level domain on the Internet. Second-level domain (SLD) is the domain name substring that is adjacent to the top-level domain. For example, edu is a SLD, which represents education organization. Third-level domain (TLD) refers to the domain name substring that is adjacent to the right of SLD. For example, lut is a TLD, which is the abbreviation of the Lanzhou University of Technology. Therefore, domain name substrings at each level have a specific meaning in its construction.

In the process of domain name resolution, when the domain name resolution request is not recorded in local network DNS servers, the resolution request is forwarded to the superior network DNS servers, until it reaches the root domain DNS servers. After reaching the root domain, the resolution requests are forwarded to the DNS servers again where the top-level, second-level, third-level, and other level domain names are located, until the domain name resolution result is found. Given that the domain name resolution request is forwarded from the superior domain to the inferior domain, if the given inferior domain name does not exist, the domain name resolution fails. Then, the domain name resolution result or the cause of the domain name resolution failure will be returned to the host that initiates the domain name resolution according to the original path of the request.

From the domain name resolution process, it is noted that the deeper level a malicious domain name is at, the greater its forwarding number is, thus the heavier burden it creates on the system. Conversely, the closer a malicious domain name is to the top-level domain, the smaller its forwarding number is, and thus the easier it can be found. In addition, because of the small quantity, short length, and high popularity of top-level domains, they are readily identified. Therefore, malicious domain names are rarely found in the top-level domain, but often exist in the second, third, or fourth domain. Hence, this study focuses on each domain name substring excluding the top-level domain.

3.2.1. Substring Statistics. The character string in the text is segmented by a sliding window with a size of *N*, and multiple substrings of length *N* are obtained, each of which is called a gram. For example, the process of 5-Gram segmentation for a string "microsoftword" of length 13 is shown in Figure 2.

When the *N*-Gram method is applied to segment the text, the value of *N* will influence the number of gained substrings. If the value of *N* is too small, the number of domain name substrings obtained by segmentation will be large, which leads to enormous calculation quantity and storage space. If the value of *N* is too large, the number of domain name substrings obtained by segmentation will be small, which can lead to little effective lexical feature information obtained by segmentation, which is not conducive to extract domain name composition structure and semantic information. After excluding the top-level domains of top 100,000 domain names in Alexa 2013, domain name substring length proportions of each level are counted, as shown in Table 1.

As seen from Table 1, after excluding top-level domains of top 100,000 domain names in Alexa 2013, the proportion of the length of each level domain name in the [3, 7] interval is up to 97.63%. Therefore, the size of the sliding window *N* is set to 3, 4, 5, 6, and 7, and each domain name excluding top-level domains is segmented by the *N*-Gram method to form the domain name substring set.

For example, after removing the top-level domain "com" of wapseo.chinaz.com, the process of second-level and third-level domains is segmented by the *N*-Gram method as shown in Figure 3. The second-level domain substring set is {chi, hin, ina, naz, chin, hina, Inaz, china, hinaz, chinaz}. And the third-level domain substring set is {wap, aps, pse, seo, waps, apse, pseo, wapse, apseo, wapseo}.

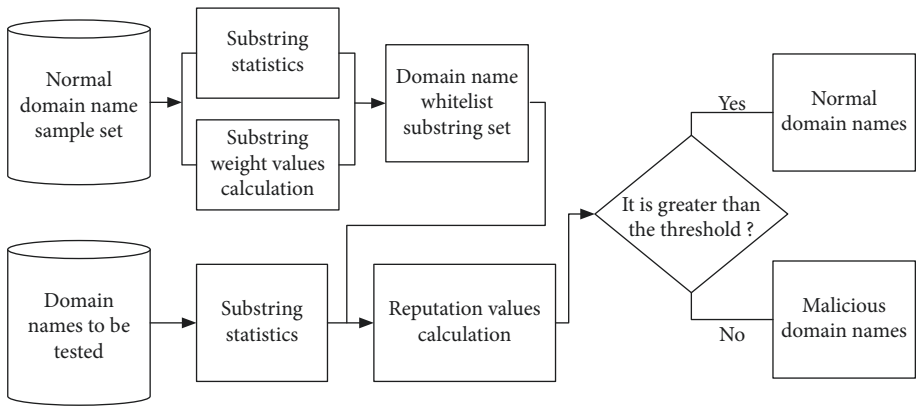


Figure 1: Flowchart of malicious domain names detection algorithm based on N-Gram.

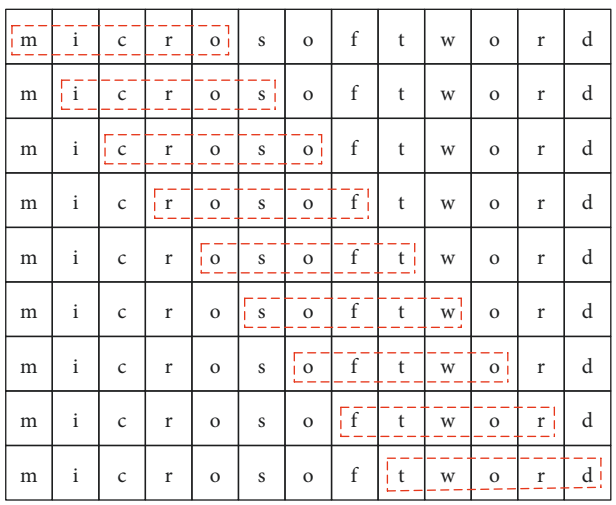


Figure 2: Principle diagram of 5-Gram segmentation.

Table 1: Each level substring length proportion.

Length	3	4	5	6	7
Proportion (%)	5.39	20.09	29.13	29.21	13.81

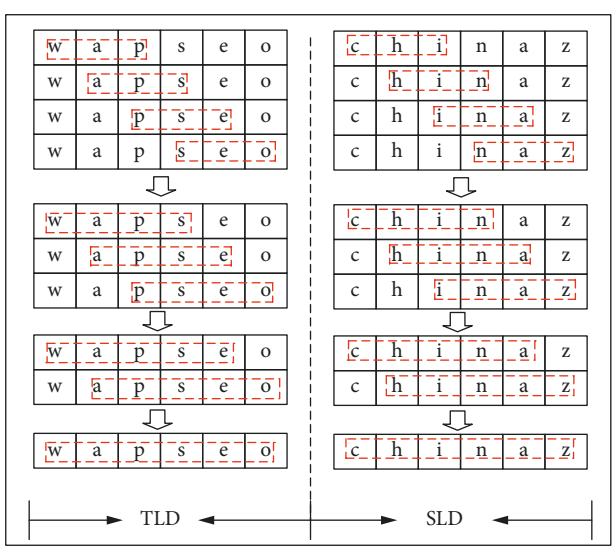


Figure 3: Process of domain name segmentation.

The top 100,000 domain names in Alexa 2013 are selected, and each domain name excluding the top-level domain is segmented into multiple domain name substrings according to its domain level with the lengths of 3, 4, 5, 6, and 7 by the N-Gram method. Furthermore, the duplicate domain name substrings are removed, and the 398,823 completely different domain name substrings are obtained as a domain name whitelist substring set. The number of domain name substrings at each domain level and of each sliding window size are calculated by the following formula:

$$\text{count}(j) = L - N + 1, \tag{1}$$

where count(j) ($j = 1, 2, \ldots, n$) represents the number of domain name substrings that are obtained from segmenting the jth-level domain of a domain name, L represents the length of jth-level domain, n represents the maximum level number of a domain name, and N represents the size of the sliding window whose value ranges from $\{N \in N^* | 3 \le N \le 7\}$.

When the size of the sliding window N is set to 3, 4, 5, 6, and 7, the number of completely different domain name substrings obtained is shown in Table 2. Where the number of domain name substrings with the length of 3 is 21,584, with the length of 4 is 84,431, with the length of 5 is 120,626, with the length of 6 is 116,908, and with the length of 7 is 55,274, with a total of 398,823 domain name substrings. And the domain name whitelist substring set is constructed according to these completely different domain name substrings.

3.2.2. Substring Weight Values Calculation. The extraction of the lexical features of the domain name turns into numerical calculation by calculating the weight values of 398,823 domain name substrings in the domain name whitelist substring set. The weight value of domain name substring is calculated by the following formula:

$$W_{N\text{-Gram}}(i) = \log_2\left(\frac{C_{N\text{-Gram}}(i)}{N}\right), \tag{2}$$

where $W_{N\text{-Gram}}$ ($N = 3$, 4, 5, 6, and 7) is the weight value of the ith domain name substring and $C_{N\text{-Gram}}(i)$ represents the

Table 2: The number of domain name substrings generated when $N = 3$, 4, 5, 6, and 7.

Sliding window	Number
3	21,584
4	84,431
5	120,626
6	116,908
7	55,274
Total	398,823

total number of the occurrences of the ith domain name substring after the top 100,000 domain names are segmented in Alexa 2013.

398,823 domain name substrings are extracted from the top 100,000 domain names in Alexa 2013 by the N-Gram method, and each domain name substring weight value is calculated. Total score of each domain name to be tested is calculated according to these domain name substring weight values. Partial domain names substring weight values from the top 100,000 domain names in Alexa 2013 are shown in Table 3.

3.3. Malicious Domain Names Detection. In the process of malicious domain names detection, a test domain name is segmented by the N-Gram method, and its reputation value is calculated according to the weight values of its substrings in the domain name whitelist substring set. Finally, the judgment of whether a domain name is malicious is made according to its reputation value. The process of the malicious domain names detection is shown in Figure 4.

3.3.1. Reputation Values Calculation. Each domain name excluding the top-level domain is segmented into multiple substrings with the lengths of 3, 4, 5, 6, and 7 by the N-Gram method. The total weight value of the domain names to be tested is calculated according to the weight values of its substrings in the domain name whitelist substring set. Additionally, the total weight value is used as the reputation value (RV) to evaluate whether the domain name to be tested is a malicious domain name. The RV is calculated by the following formula:

$$\mathrm{RV}(l) = \sum_{i=1}^{m} W_{N-Gram}(i), \tag{3}$$

where $W_{N\text{-Gram}}$ ($N = 3$, 4, 5, 6, and 7) is the weight value of ith domain name substring which is referenced from 398,823 domain name substring weight values (as shown in Table 3), $l\{l \in N^* | l > 0\}$ represents a domain name to be tested that the serial number is l, and m represents the total number of obtained domain name substrings whose domain name l is segmented, when the sliding window N is set to 3, 4, 5, 6, and 7. Since the substrings of the malicious domain names appear less frequently in the domain name whitelist substring set, the RV of malicious domain names is smaller. On the contrary, the RV of normal domain names is larger. Therefore, a simple technique of thresholding can be used to achieve the detection of malicious domain names.

Table 3: Partial domain names substring weight values from the top 100,000 domain names in Alexa 2013.

N-Gram	$C_{N\text{-Gram}}(i)$	$W_{N\text{-Gram}}(i)$
ing	3139	10.031
ter	2105	9.454
line	1270	8.310
blog	1194	8.221
direc	587	6.875
forum	452	6.498
ectory	341	5.828
ogspot	293	5.609
rectory	341	5.606
youtube	220	4.974
rketing	167	4.576

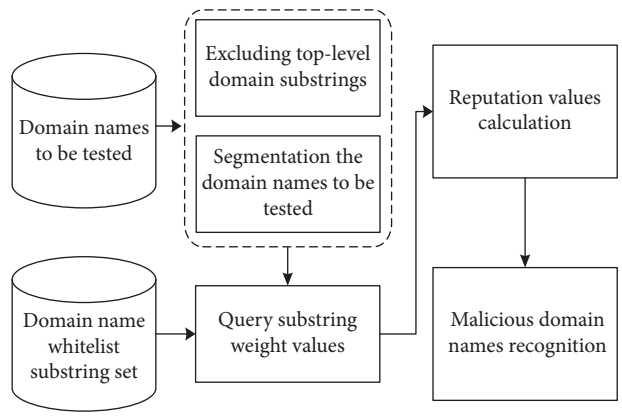


Figure 4: Flow of malicious domain names detection.

For example, after removing the top-level domain "com" and replacing the letter "o" in the normal domain name "taobao.com" with the number "0," the RV of the normal domain name "taobao.com" and the malicious domain name "ta0ba0.com," which are similar to the normal domain name "taobao.com," can be calculated. The RV is calculated as follows:

$$\begin{aligned}
\mathrm{RV}_{\mathrm{taobao}} &= W_{\mathrm{tao}} + W_{\mathrm{aob}} + W_{\mathrm{oba}} + W_{\mathrm{bao}} + W_{\mathrm{taob}} \\
&\quad + W_{\mathrm{aoba}} + W_{\mathrm{obao}} + W_{\mathrm{taoba}} + W_{\mathrm{aobao}} + W_{\mathrm{taobao}} \\
&= 2.736 + 3.807 + 2.321 + 3.222 + 0.807 \\
&\quad + 1.459 + 1.321 + 0.485 + 0.847 + 0.222 \\
&= 17.227, \\
RV_{\mathrm{ta0ba0}} &= W_{\mathrm{ta0}} + W_{\mathrm{a0b}} + W_{\mathrm{0ba}} + W_{\mathrm{ba0}} + W_{\mathrm{ta0b}} \\
&\quad + W_{\mathrm{a0ba}} + W_{\mathrm{0ba0}} + W_{\mathrm{ta0ba}} + W_{\mathrm{a0ba0}} + W_{\mathrm{ta0ba0}} \\
&= 0 + 0 + 0.415 + 0 + 0 + 0 + 0 + 0 + 0 \\
&\quad + 0 + 0 \\
&= 0.415.
\end{aligned} \tag{4}$$

By calculating the RV of the normal domain name "taobao.com" and the malicious domain name "ta0ba0.com," it can be seen that the RV of the normal domain name "taobao.com" is 17.227. When the size of N is 3, 4, 5, 6, and 7,

the normal domain name "taobao.com" is segmented into multiple domain name substrings which appear frequently in the domain name whitelist substring set. However, the RV of the malicious domain name "ta0ba0.com" is 0.415. When the size of N is 3, 4, 5, 6, and 7, the malicious domain name "ta0ba0" is segmented into multiple domain name substrings which appear with very small probability in the domain name whitelist substring set.

3.3.2. Threshold Setting for Malicious Domain Names Detection. In the process of malicious domain names recognition, the size of threshold decides the accuracy rate of the detection algorithm in this study. In order to attain the superior detection accuracy rate, the variable parameter threshold D is debugged on the same dataset and the optimal threshold D is selected. The corresponding relationship between the threshold D size and the detection accuracy rate is shown in Figure 5.

From Figure 5, it can be seen that the curve of detection accuracy rate shows a trend of left rising and right falling. When threshold D is 0.65, the detection accuracy rate reaches the optimal level of 94.04%, which refers to it as a better detection effect when the threshold D is 0.65. Therefore, threshold D in this study is set to 0.65.

In this study, the threshold for malicious domain names detection is set on the basis of the domain name whitelist substring set that is constructed by the top 100,000 domain names in Alexa 2013. If the domain name whitelist sample on constructing domain name whitelist substring set is replaced, the threshold for malicious domain names detection needs to be reset according to the above steps.

When the threshold is set, the RV of the domain name to be tested is calculated to judge whether the domain name to be tested is malicious based on the size of the RV and threshold for malicious domain names detection. If the RV of the domain name to be tested is greater or equal than the threshold for malicious domain names detection, the domain name is judged to be a normal domain name. If not, it is a malicious domain name.

4. Experimental and Result Analysis

To verify the performance of the proposed algorithm based on N-Gram, experiments on malicious domain names detection were conducted using large volumes of data collected and published by Alexa and other sources.

4.1. Experiments. The experimental environment is shown in Table 4.

4.2. Experimental Data

4.2.1. Domain Name Whitelist Substring Set. The top 100,000 domain names in Alexa 2013 are selected as the domain name whitelist sample set. Each domain name excluding the top-level domain is segmented into multiple domain name substrings according to its domain level with the lengths of 3, 4, 5, 6, and 7 by the N-Gram method. 398,823 completely different domain name substrings are extracted as the domain name whitelist substring set.

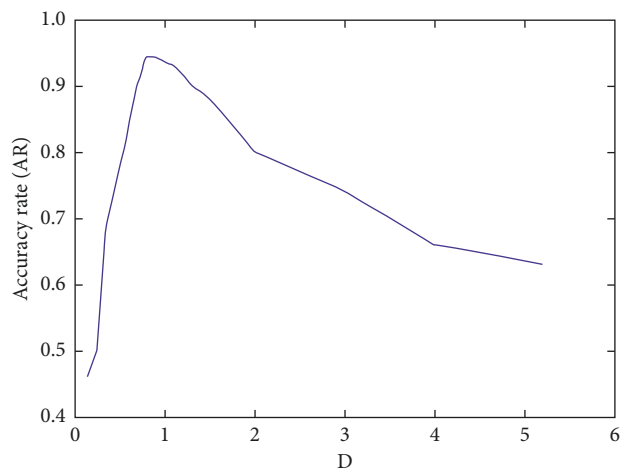


Figure 5: Relationship curves of accuracy rate with D.

Table 4: Experimental environment.

Parameters	Value
CPU	AMD A12-9700 2.5 GHZ
GPU	AMD R8 M435DX
Memory	8 GB
OS	64-bit Windows 10
Platform	Jupyter Notebook
Python	3.5

4.2.2. Domain Name Sample Set to Be Tested. In this study, 10,265 domain names from the Alexa 2017 and Malware domain list are collected and collated [30, 31]. The 8,000 domain names with the highest number of accesses in Alexa 2017 are taken as the normal domain name sample set, and the 2,265 domain names in Malware domain list (malicious domain names that are generated by the DGA [32], botnet [33], Conficker [34], and Spam [35]) are taken as the test sample set of malicious domain names.

4.3. Evaluation Standard. In order to evaluate the performance of the malicious domain names detection algorithm based on N-Gram in malicious domain names detection, the accuracy rate (AR), false negative rate (FNR), and false positive rate (FPR) are used. The evaluation standard is calculated based on the confusion matrix [36] of the experimental results, as shown in Table 5. Evaluation standard is calculated by the following formula:

$$\begin{aligned} \mathrm{AR} &= \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{FP}+\mathrm{TN}+\mathrm{FN}} \times 100\%, \\ \mathrm{FNR} &= \frac{\mathrm{FN}}{\mathrm{TP}+\mathrm{FN}} \times 100\%, \\ \mathrm{FPR} &= \frac{\mathrm{FP}}{\mathrm{TN}+\mathrm{FP}} \times 100\%, \end{aligned} \tag{5}$$

Table 5: Confusion matrix of TN, FN, FP, and TP.

Actual	Predicted	
	Negative	Positive
Negative	True negative (TN)	False positive (FP)
Positive	False negative (FN)	True positive (TP)

where TN represents the number of normal domain names that are correctly detected by the algorithm, FN represents the number of malicious domain names that are incorrectly reported as normal domain names, FP represents the number of normal domain names that are incorrectly reported as malicious domain names, and TP represents the number of malicious domain names that are correctly detected by the algorithm.

4.4. Result Analysis. Figure 6 shows the effect of threshold D on false negative rate and false positive rate, where Figure 6(a) shows the relation between the threshold D on the X-axis and the false negative rate on the Y-axis and Figure 6(b) shows the relation between the threshold D on the X-axis and the false positive rate on the Y-axis.

From the results of the two experiments presented in Figure 6, when the threshold D is less than 0.65 and gradually increases within this range, the curve of detection accuracy rate takes on ascend trend (Figure 5). The main reason is that the number of malicious domain names accurately detected by the algorithm increases and the number of malicious domain names incorrectly reported as normal domain names gradually decreases, which makes the detection accuracy rate increase gradually and the false negative rate decrease gradually (Figure 6(a)). When the threshold D is greater than 0.65 and it gradually increases within this range, the curve of detection accuracy rate takes on downward trend (Figure 5). The reason is that the number of normal domain names accurately detected by the algorithm decreases and the number of normal domain names incorrectly reported as malicious domain names gradually increases, which makes the detection accuracy rate decrease gradually and the false positive rate increase gradually (Figure 6(b)).

4.5. Comparison with Other Approaches. To verify the effectiveness of the proposed algorithm, experiments were also performed using the methods in the latest literatures [8, 10, 13, 14] with the same experimental conditions. Accuracy and computational cost are the measures for the effectiveness of the algorithms. Performance comparisons in terms of AR, FNR, FPR, and time overhead (TO) are shown in Table 6.

Our proposed method yielded a superior combinational result of accuracy rate and computational efficiency. Other methods either has lower accuracy rate or is computational more expensive. In addition, compared to the other methods that are based on machine learning techniques, our method is much easier to add new data when they become available. While the machine learning algorithms require a new training process of all the data, our method only needs modifications to the weight of relevant substrings.

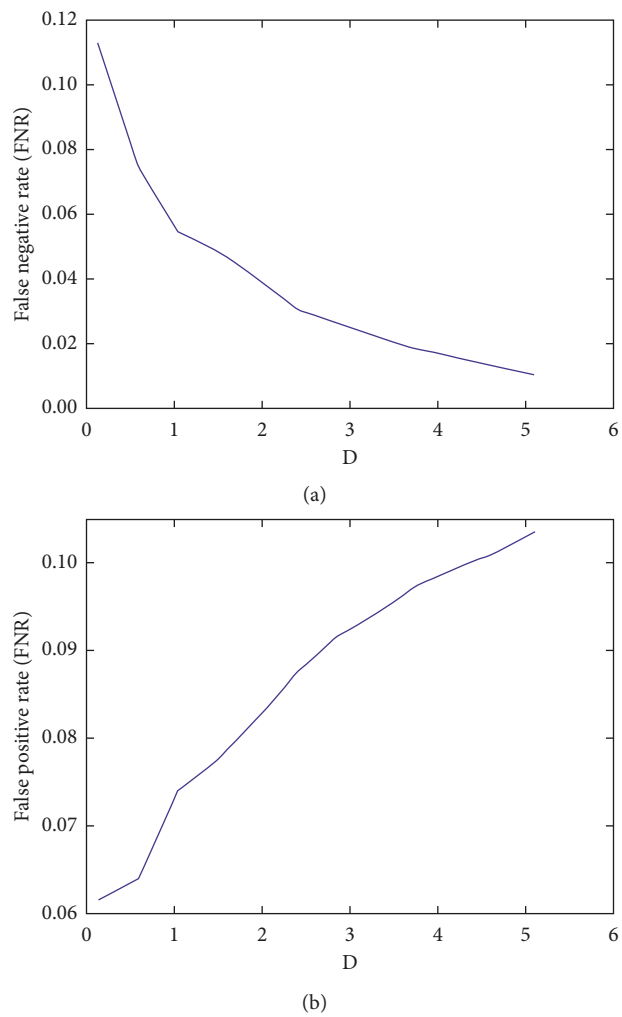


Figure 6: Threshold D effect on (a) false negative rate curve and (b) false positive rate curve.

Table 6: The performance comparison between our approach and methods in [8, 10, 13, 14].

Method	AR (%)	FNR (%)	FPR (%)	TO (s)
Shi et al. [8]	95.75	4.29	6.62	42.68
Ma et al. [10]	91.04	2.65	7.11	18.92
Wu et al. [13]	91.52	8.48	1.50	32.08
Song et al. [14]	93.47	4.42	7.43	38.27
Our approach	94.04	7.42	6.14	31.75

5. Conclusion and Future Work

This study proposes a new method for malicious domain names detection. The main contributions are as follows:

(1) The top 100,000 domain names in Alexa 2013 are taken as the domain name whitelist sample set to construct the domain name whitelist substring set

(2) The N-Gram method in the natural language processing technology is used to segment the domain names, and the malicious domain names are quickly recognized according to its occurrence number in the substring set

Compared to the malicious domain names detection methods proposed by [8, 10, 13, 14], our approach demonstrated a superior comprehensive performance of lower time overhead and higher detection accuracy rate. It has a good practical value in defending against the botnet, Spam, and remote access Trojan attack and can help security experts and organizations in their fight against cybercrime. However, our proposed approach is not comprehensive in homographic domain names (such as linkedin.com and 1inkedin.com, apple.com and app1e.com) detection aspects, but if the malicious domain names are generated randomly, our approach can detect them efficiently. These homographic domain names detection problems should be considered in the future.

Data Availability

The Alexa and Malware domain list datasets used to support the findings of this study are cited at relevant places within the text as references [28, 30, 31, 33–35].

Conflicts of Interest

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

Acknowledgments

This research work was supported by the National Science Foundation of China under Grant nos. 51668043 and 61262016, the CERNET Innovation Project under Grant nos. NGII20160311 and NGII20160112, and the Gansu Science Foundation of China under Grant nos. 18JR3RA156.

References

[1] National Internet Emergency Center, "36th internet security threat report," 2018, http://www.cert.org.cn/publish/main/44/index.html.

[2] W. Quan, C. Xu, J. Guan, H. Zhang, and L. A. Grieco, "Scalable name lookup with adaptive prefix bloom filter for named data networking," *IEEE Communications Letters*, vol. 18, no. 1, pp. 102–105, 2014.

[3] S. Yadav, A. K. K. Reddy, A. L. N. Reddy, and S. Ranjan, "Detecting algorithmically generated domain-flux attacks with DNS traffic analysis," *IEEE/ACM Transactions on Networking*, vol. 20, no. 5, pp. 1663–1677, 2012.

[4] W. Quan, C. Xu, A. Vasilakos, and J. Guan, "TB2F: tree-bitmap and bloom-filter for a scalable and efficient name lookup in content-centric networking," in *Proceedings of the IFIP Networking Conference*, pp. 1–9, Trondheim, Norway, June 2014.

[5] L. Bilge, S. Sen, D. Balzarotti, E. Kirda, and C. Kruegel, "Exposure," *Acm Transactions on Information and System Security*, vol. 16, no. 4, pp. 1–28, 2014.

[6] R. Sharifnya and M. Abadi, "DFBotKiller: DFBotKiller: domain-flux botnet detection based on the history of group activities and failures in DNS traffic," *Digital Investigation*, vol. 12, no. 12, pp. 15–26, 2015.

[7] B. Yu, L. Smith, M. Threefoot, and F. Olumofin, "Behavior analysis based DNS tunneling detection and classification with big data technologies," in *Proceedings of International Conference on Internet of Things and Big Date*, pp. 284–290, Rome, Italy, April 2016.

[8] Y. Shi, G. Chen, and J. Li, "Malicious domain name detection based on extreme machine learning," *Neural Processing Letters*, vol. 48, no. 3, pp. 1347–1357, 2017.

[9] S. Tian, C. Fang, J. Liu, and Z. Lei, "Detecting malicious domains by massive DNS traffic data analysis," in *Proceedings of the 8th International Conference on Intelligent Human-Machine Systems and Cybernetics*, pp. 130–133, Zhejiang, China, August 2016.

[10] Z. Ma, H. Chen, J. Yang, and X. L., "Novel network intrusion detection method based on IPSO-SVM algorithm," *Computer Science*, vol. 45, no. 2, pp. 231–235, 2018.

[11] P. Kintis, N. Miramirkhani, C. Lever, and Y. Chen, "Hiding in plain sight: a longitudinal study of combosquatting abuse," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, Dallas, TX, USA, August 2017.

[12] P. Zhang, T. Liu, Y. Zhang, J. Ya, and J. Shi, "Domain watcher: detecting malicious domains based on local and global textual features," in *Proceedings of the International Conference on Computational Science*, pp. 2408–2412, Zurich, Switzerland, June 2017.

[13] Z. Wu, J. Zhang, M. Yue, and C. Zhang, "Approach of detecting low-rate DoS attack based on combined features," *Journal on Communications*, vol. 38, no. 5, pp. 19–30, 2017.

[14] W. Song and B. Li, "A method to detect machine generated domain names based on random forest algorithm," in *Proceedings of the International Conference on Information System and Artificial Intelligence*, pp. 509–513, Hong Kong, China, June 2017.

[15] C. Xiong, P. Li, P. Zhang, Q. Liu, and J. Tan, "MIRD: trigram-based malicious URL detection implanted with random domain name recognition," in *Proceedings of the 6th International Conference on Applications and Techniques in Information Security*, pp. 303–314, Beijing, China, November 2015.

[16] D. Truong, G. Cheng, A. Jakalan, X. Guo, and A. Zhou, "Detecting DGA-based botnet with DNS traffic analysis in monitored network," *Journal Of Internet Technology*, vol. 17, no. 2, pp. 217–230, 2016.

[17] X. Zang, J. Gong, and X. Hu, "Detecting malicious domain name based on AGD," *Journal on Communications*, vol. 39, no. 7, pp. 15–25, 2018.

[18] R. Sharifnya and M. Abadi, "A novel reputation system to detect DGA-based botnets," in *Proceedings of the 3rd International Conference on Computer and Knowledge Engineering*, pp. 417–423, Mashhad, Iran, October 2013.

[19] J. Kwon, J. Lee, H. Lee, and A. Perrig, "PsyBoG: PsyBoG: a scalable botnet detection method for large-scale DNS traffic," *Computer Networks*, vol. 97, pp. 48–73, 2016.

[20] Y. Zhang, Y. Lu, and Y. Zhang, "Detecting domain flux through patterns of domain names' alphanumeric characters and querying behavior of hosts," *Journal of Xian Jiaotong University*, vol. 47, no. 8, pp. 54–60, 2013.

[21] R. Vinayakumar, K. Soman, and P. Poornachandran, "Detecting malicious domain names using deep learning approaches at scale," in *Proceedings of the 3rd International Symposium on Intelligent Systems Technologies and Applications*, pp. 1355–1367, Manipal, India, September 2017.

[22] S. Yadav, A. K. K. Reddy, A. L. N. Reddy, and S. Ranjan, "Detecting algorithmically generated malicious domain names," in *Proceedings of the 10th ACM Sigcomm Conference on Internet Measurement*, pp. 48–61, Melbourne, Australia, November 2010.

[23] K. Huang, J. Fu, J. Huang, and P. Li, "A malicious domain detection approach based on characters and resolution features," *Computer Simulation*, vol. 35, no. 3, pp. 287–292, 2018.
[24] W. Zhang, J. Gong, X. Liu, and X. Hu, "Lightweight domain name detection algorithm based on morpheme features," *Journal of Software*, vol. 27, no. 9, pp. 2348–2364, 2016.
[25] Y. Zhang, Y. Zhang, and J. Xiao, "Detecting the DGA-based malicious domain names," in *Proceedings of the International Standard Conference on Trustworthy Computing and Services*, pp. 130–137, Beijing, China, November 2013.
[26] L. Zhao, "Research on a high efficiency pattern matching algorithm for intrusion detection," *Computer and Digital Engineering*, vol. 45, no. 8, pp. 1592–1596, 2017.
[27] D. A. Orr and L. Sanchez, "Alexa, did you get that? Determining the evidentiary value of data stored by the Amazon Echo," *Digital Investigation*, vol. 24, pp. 72–78, 2018.
[28] Alexa top global sites, 2013, http://www.alexa.com/topsites.
[29] E. Casalicchio, M. Caselli, and A. Coletta, "Measuring the global domain name system," *IEEE Network*, vol. 27, no. 1, pp. 25–31, 2013.
[30] Alexa top global sites, 2017, http://www.alexa.cn/siterank/14.
[31] Malware domain list, 2017, http://www.malwaredomainlist.com.
[32] S. Schüppen, D. Teubert, P. Herrmann, and U. Meyer, "FANCI: feature-based automated NX- Domain classification and intelligence," in *Proceedings of the 27th USENIX Security Symposium*, pp. 1165–1181, Baltimore, MD, USA, August 2018.
[33] DNS-BH malware domain blacklist, 2016, http://www.malwaredomains.com.
[34] Phish Tank, 2013, http://www.phishtank.com.
[35] Blacklist provided by joewein.net (JWSDB), 2015, http://joewein.net/spam/blac-klist.htm.
[36] A. Aborujian and S. Musa, "Cloud-based DDOS http attack detection using covariance matrix approach," *Journal of Computer Networks and Communications*, vol. 2017, Article ID 7674594, 8 pages, 2017.