Computer Science and Information Technologies
Vol. 7, No. 2, July 2026, pp. 141~158
ISSN: 2722-3221, DOI: 10.11591/csit.v7i2.p141-158

Improving Botnet host prediction with encryption and GRU for enhanced network security

Omega Joel Patria Moata, Irwansyah Saputra
Department of Computer Science, Nusa Mandiri University, Jakarta, Indonesia

Article Info

Article history:

Received Jun 27, 2025
Revised Aug 28, 2025
Accepted Nov 28, 2025

Keywords:

Botnet detection
Deep learning optimization
Encryption
Intrusion detection system
IP address analysis
Network security

ABSTRACT

This paper examines the challenges of reliably and securely predicting Botnet hosts, a crucial aspect of network security. Existing Botnet detection systems often fail to address data privacy concerns and struggle with evolving attack methods. This study proposes an innovative approach to improve the security and accuracy of Botnet host prediction by integrating deep learning with encryption. The proposed method employs encryption techniques such as data encryption standard (DES) and blum-blum-shub (BBS) to protect sensitive data in a text data set of 2,100 IP addresses, consisting of Botnet hosts and benign hosts. Several pre-processing techniques, including moving average and missing value handling, are implemented to optimize the model performance. The effectiveness of the system is evaluated using performance metrics such as F1-score, recall, accuracy, and precision. Experimental results show that the proposed approach significantly outperforms existing methods in accuracy, which have not achieved the maximum accuracy per IP Host within a given time frame, while providing enhanced security through encryption on text data. The study concludes that combining deep learning with encryption on text data offers a promising solution for reliable and secure Botnet host prediction data. Future research will focus on testing larger and more diverse data sets, as well as analyzing the impact of different encryption techniques on the overall accuracy and security of the system.

Corresponding Author:

Omega Joel Patria Moata
Department of Computer Science, Nusa Mandiri University
Jakarta, Indonesia
Email: 14230016@nusamandiri.ac.id

1. INTRODUCTION

This study builds on the work of Oluwaseye *et al.* [1], who explored malware detection in botnets using deep learning. However, their study did not integrate encryption or information security into the deep learning model. To expand the capabilities of Botnet detection, this research incorporates text encryption techniques, data preprocessing methods, deep learning optimization strategies, and overfitting prevention techniques. The proposed approach is designed to improve both the learning efficiency and predictive accuracy of a secure Botnet host prediction model.

The threat of cybercrime, particularly Botnets, continues to increase globally. In Indonesia, cyber losses are estimated at $895 billion, representing 1.20% of total global losses of $71.62 trillion according to DAKA Advisory [2]. The urgency of information system security is further amplified in the era of Society 5.0, where technologies such as the internet of things (IoT) and artificial intelligence (AI) play a vital role [3]. Unfortunately, many AI-based prediction systems in sectors such as education, government, and healthcare have not implemented encryption, leaving sensitive personal information vulnerable to theft and

misuse. This vulnerability is exemplified by the massive data leaks carried out by the hacker “Bjorka,” which exposed sensitive data of Indonesian citizens, including NIK, KK numbers, and telephone numbers, allegedly obtained from institutions such as Indihome, KPU, and SIM registration databases [4]. Such incidents highlight the fragility of national digital infrastructure and the urgent need for proactive solutions to protect sensitive information, particularly IP addresses used in botnet host identification. Globally, ransomware and phishing remain dominant threats, accounting for more than 30.25% of cyberattacks in recent decades. These evolving threats demand the development of adaptive and secure detection systems. Figure 1 illustrates global cybercrime activity based on real-time intrusion detection data, emphasizing the scale and persistence of these threats.

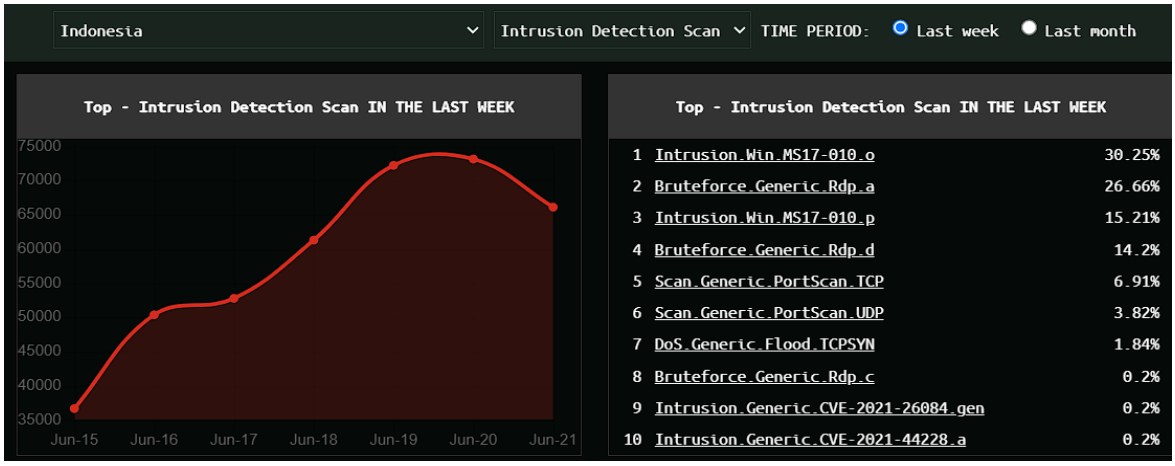

Figure 1. Data source (https://cybermap.kaspersky.com/stats#country=144&type=IDS&period=w)

The ransomware issue that can be detected from the intrusion detection scan system in the last few weeks and phishing also continue to dominate the cyber threat landscape, demanding the development of a more adaptive and secure detection system. Around more than 30.25% of cyber threats recorded in the last few decades.

To overcome the security gap and improve accuracy, this study implements the data encryption standard (DES) [5] and blum-blum-shub (BBS) [6] encryption algorithms. DES, as a symmetric block cipher, is widely used for encrypting personal data, while BBS enhances randomness and security in key generation. Recent feasibility testing confirms that the improved DES-BBS method significantly strengthens encryption security while maintaining efficiency, with average processing times of only 2 seconds. Table 1 presents examples of encryption and decryption results, demonstrating the robustness and efficiency of the proposed approach.

Table 1. Encryption decryption text (https://doi.org/10.63447/jimik.v6i2.1385)

No	Plaint text	Chiper text	Time process
1	Great! Your work is very good and detailed. I really appreciate your effort and dedication in completing this task.	uuzjEgoB14nx+TF6oHvR1KbmYfOjGnegVgIfR9jz9MiPuK7Ucylu/mjaWKGLyXKLKqYK+dcDJ43caWoDJb6U7sxRd3aHIyV3ze4nUsyMi7nwWFtxVT7X8nVH+gd3bI4LAZZkQ0oOS8XL6pStvmJQ+WF6mzlKZWFXLfMAVFfH2FU=	5 ms
2	Perhaps we could consider some minor improvements to the user interface to make it more user-friendly. I believe it would improve the overall user experience.	tOXPbTGUcM5n4yBDaEbLYki6xnJO9ICCUB3BazApsR4C+MR4y4tUfXFoKl7okFvk73dU3a/1eO3k01+qtoRdsyBHnA47wXEHc28WCw88LJdnopR2fbvftm8i0tfMytlMqHi73s4VXwWxPOLqlPLxFiHfICAX82QTEYTd1fQ//D0gAYk4zUnnoWQh5JA3NEbBkkxXiQO9Cs=	2 ms
3	Your presentation is already good, but try to speak at a slower pace and engage more with the audience. This will make your presentation more memorable and memorable	XdwJ6l074HyNnlaInfpIEXLuhGQ8RHmKpkNX7QxGvQ1ioJH1NHsnkq3rOUb3bV4ISwcze+Yr9SuSXA1aE8W00ZU9PhLfBVk3wOTRHnaJ2PAFrgXoAbsj68xzlH9u7CCOI43Drk/9s2pZHc3ZC2dWOcZjGhzGr7dY+zUTxtFVYjiYqoLur4+U7sWCfbBLek2jcyYq/TN1n7WbDY7Vkc7OzQ==	1 ms

Building on these foundations, this study integrates encryption with machine learning techniques to improve the prediction of networks or IP addresses suspected of being Botnets. The main contribution is the integration of DES and BBS encryption with a deep learning model based on Gated Recurrent Units (GRU) for Botnet host prediction. The proposed model demonstrates significant performance improvements, achieving high accuracy on test data while maintaining realistic adaptability to dynamic network conditions. To further enhance reliability, preprocessing techniques such as outlier handling [7], missing data treatment [8], and moving averages are applied. Model efficiency is optimized using the Adam optimizer and early stopping, ensuring robust performance without overfitting.

Based on these considerations, this study addresses three key research questions: i) How can a Botnet host prediction system be designed and optimized? ii) How does the encryption algorithm affect the security of text data in the prediction system? and iii) How effective are the Adam optimizer and early stopping techniques in improving prediction performance? Previous studies have not explicitly addressed the integration of encryption into deep learning-based Botnet prediction, making this research an important contribution.

The significance of this work lies in its dual focus on accuracy and security. By integrating encryption with GRU-based prediction, the study provides a secure and efficient framework for Botnet host detection. This approach not only strengthens data protection but also enhances predictive performance, offering practical benefits for governments, enterprises, and service providers. Ultimately, the proposed system contributes to building resilient cybersecurity infrastructures in the era of Society 5.0, where interconnected IoT and AI systems are increasingly vulnerable to exploitation.

2. RELEATED WORKS

A basic understanding of the principles of data encryption and basic techniques of machine learning and standard encryption are the foundation of this research. In this paper, Botnets will be detected based on the characteristics of network traffic and its flow with machine learning. Existing Botnet detection systems are mainly based on intrusion detection and identification of command and control (C&C) servers using natural language processing (NLP) techniques. The main limitation of current models is that the name of the C&C server can change frequently using various name-changing algorithms and intrusion signatures. This research focuses on Botnet detection based on network traffic. An alternative approach in developing this machine learning model also involves encryption to maintain information security and secure the extraction of results from each operation, including correlation results, entropy results, and model evaluation results [9].

2.1. Botnet

Botnet is a network of "bots," which is a collection of interconnected networks managed by a single entity known as a "bot-herder." If an attacker intends to target a large organization such as Google or Amazon, they will need significant computing resources to penetrate a web server's security firewall. Essentially, attackers form an army using their malware network. They initially target individual users through hacking methods such as phishing or malware attacks, gaining command and control over their devices. The attacker then takes over all the compromised devices, forming them into a network. With the help of this network, the attacker can launch denial-of-service attacks or brute-force attacks, which can disrupt network traffic or damage the victim's network servers [10]. The network traffic mechanism is illustrated in Figure 2.

2.2. Data preprocessing

The data pre-processing steps, illustrated in the fishbone diagram in Figure 3, start with the cs448b_ipasn dataset which contains Botnet-related information. The initial dataset statistics show a mean of 48%, a median of 51%, and a mode of 0%. Several pre-processing techniques are applied, starting with outlier handling to detect and remove extreme values that may interfere with the modeling process [7]. Missing data handling is also performed to address gaps in the dataset [8]. In addition, the moving average technique is used to smooth out data fluctuations and highlight general trends, with visualization through line graphs. As a widely used technique in time series analysis, moving averages help summarize overall data patterns efficiently [10]. After pre-processing, the characteristics of the dataset change to a mean of 28.2%, a median of 29.9%, and a mode of 49.9%, revealing an anomalous network with a repeating pattern, which is a new aspect adapted from the reference literature. To improve the predictive capability, using recurrent neural networks (RNN) are used to analyze the 30-day forward prediction period by comparing the "Local IP Flow" values across days for ten different IPs. This approach helps identify daily patterns in network activity. Studies have shown that long short-term memory (LSTM) networks outperform traditional methods and baseline RNN models, especially for long-term forecasts. For example, LSTM models have been used to predict lake water levels with 78% higher accuracy than the Naïve Method for 60-day forecasts.

Additionally, GRU have been shown to be effective for longer prediction periods, such as 120-day water level forecasts, where GRU outperforms both RNN and LSTM in performance metrics. Due to their faster training time, GRUs are well-suited for real-time hydrological predictions that require frequent model updates. Hybrid models, which integrate RNNs, LSTMs, and GRUs with other statistical and deep learning techniques such as LSTM-CNN, further improve the accuracy and efficiency of predictions [10].

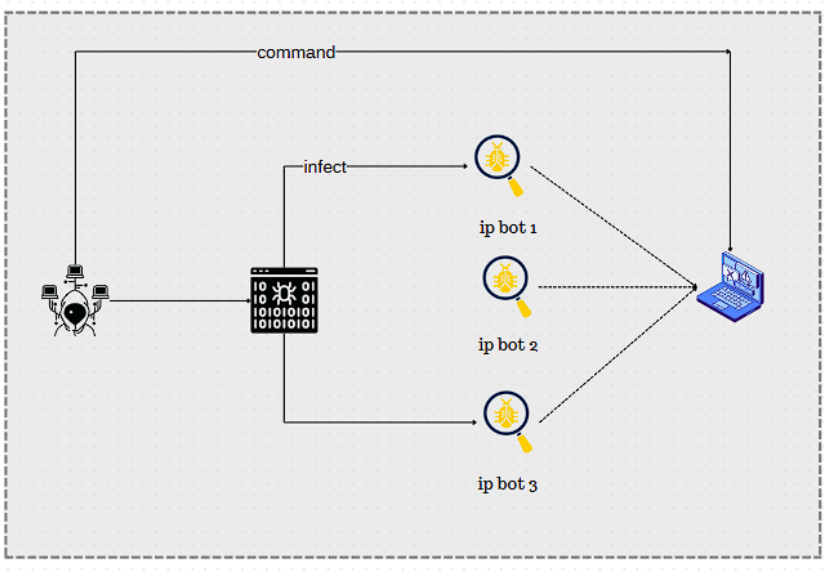

Figure 2. Illustration of general bot network architecture

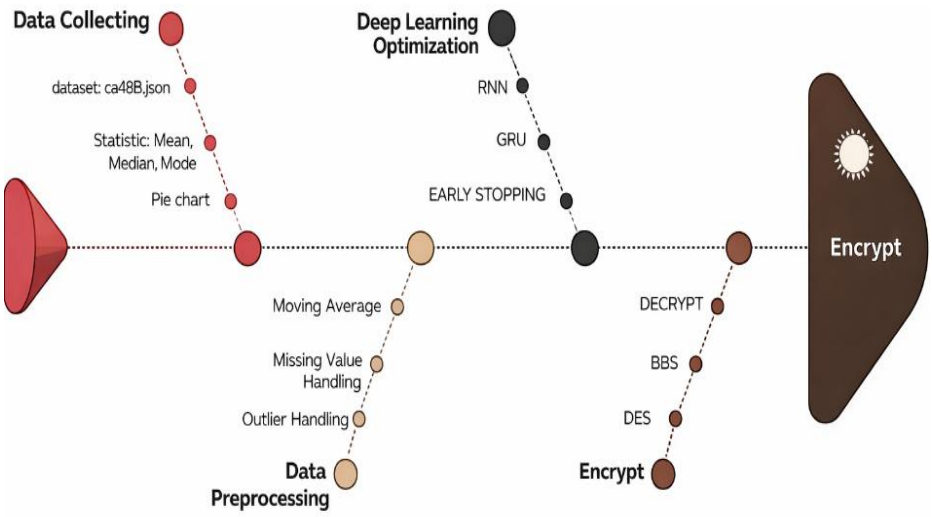

Figure 3. Fishbone diagram of encryption Botnet, GRU

RNN processes the input sequence X1, X2 ..... Xt. At each time step "t", the network updates its hidden state "h", by taking into account the input of the current time step "x," and the hidden state from the time step "h" in Figure 4. This hidden state is calculated using (1):

$$h1 = \sigma(wh \times h1 + Wx \times xt + bh) \quad (1)$$

where “h” is the hidden state at current time step “t”, h is the hidden time step of the previous time step “t-1” x is the input at current time span “t”, w is the weight matrix associated with the hidden state, w is the weight matrix for the input. hidden state, w is the weight matrix for the input. bh is the bias vector “ϕ” is a non-linear activation function, such as tanh function or ReLU, which introduces non-linearity to the network and helps it learn complex patterns. complex patterns. The output at each time step “y” is calculated using the current hidden state “ht” with (2):

$$y_{t=}\,\phi\,(w_y \times h_t + b_y) \tag{2}$$

“y” output step at time “t”, “wy” is the weight matrix for the output layer t “by” is the bias vector for the output and ‘ϕ’ as the activation function used in the output layer. The RNN is trained by minimizing a loss function L over the entire sequence, calculated as the sum of the individual losses at each time step in (3):

$$L = \sum_{i=1}^{T} \quad Lt(y_t \underline{yt}) \tag{3}$$

Where “L” represents the loss at time step “t,” and “yt” represents the anticipated output at time step “t.” The actual target output y at time step “t,” where “t” and “T” represent the total number of time steps of the sequence. The gated GRU architecture is shown. They are xt and yt for input and output, rt and zt for reset and update gates, and ht and ht for activation and candidate activation, respectively. The two gates [10] have the following definitions in (4):

$$Z_t = \sigma\,(\,W_z X_t + U_z h_{t-1} + b_z) \tag{4}$$

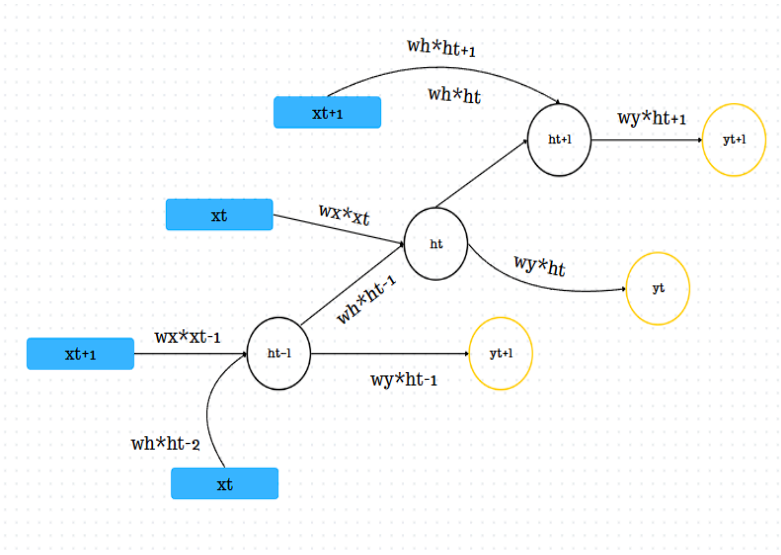

Figure 4. Flow diagram of RNN

Figure 5 is a more in-depth interpretation of the group architecture. Update gate "Z" determines which bits of the long-term state should be deleted and which should be added. Which part of the previous state is shown to the main layer or activation candidate "h" is determined by the reset gate "rtt." The GRU model can therefore be expressed in the following way:

$$R_t = \sigma\,(W_r X_t + U_h h_{t-1} + b_r \tag{5}$$

$$-h_t = tanh\,(W_h X_t + u_h(r_t . h_{t-1}) + b_h) \tag{6}$$

$$h_t = (1 - z_t).h_{t-1} + Z_t . \sim h_t \tag{7}$$

where candidate activity “ht” analyzes the input vector “xt” and the previous short-term state “ht−1” and only the most relevant part is stored in the vector “ht”. This vector “ht” represents the output state of the model.

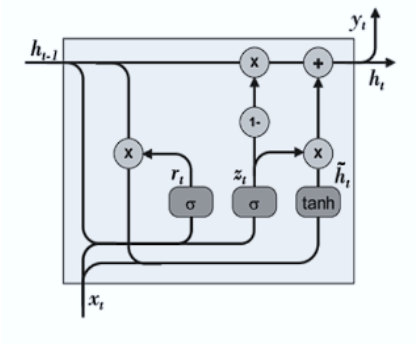

Figure 5. Architecture of GRU cell

2.3. Encryption

2.3.1. Cryptography

Cryptography serves as a method for individuals to communicate secret messages securely with each other or with their intended recipients. It involves the use of a code system to ensure that the message remains incomprehensible to third parties or individuals with malicious intent, such as hackers. This security measure becomes especially important when the transmission system is vulnerable to interception by unauthorized parties. Cryptography serves as a technique for preserving and transmitting data in a format that can only be understood by those with the proper decryption key. understood by those with the proper decryption key, ensuring the protection of critical information whether it is stored on various storage media or transmitted over a communications network [11]. Cryptography is the science and engineering used to protect data from unauthorized third parties. The two fundamental processes in cryptography are encryption and decryption. Encryption involves transforming plain text into cipher text using a key. In contrast, decryption involves converting the ciphertext back to the original plaintext using the same key. This process is particularly relevant when using symmetric key cryptography, where the same key is used for both encryption and decryption [12]. In almost all cryptographic applications, the use of pseudorandom numbers is very common. These numbers exhibit characteristics that simulate true randomness, making it difficult for malicious manipulation of the application. randomness, making it difficult for manipulation. The algorithm responsible for generating these pseudorandom numbers, which are essentially sequences of bits, is referred to as a pseudorandom number generator (PRNG) [13]. A PRNG is a deterministic algorithm designed to generate sequences that successfully pass various randomness tests. These generators often use recursion to generate sequences of numbers that mimic randomness. In this process, numbers are generated recursively, with each subsequent number in the sequence determined by the previous number. The iteration begins with a specific value known as the seed. Importantly, when the repetition starts with the same seed, it will produce the same sequence. The length of the sequence before the repetition is called the period or cycle length [14]. From the encryption of the use of the DES and BBS methods.

2.3.2. Data encryption standard

The DES algorithm is a subset of symmetric cryptography, which specifically falls under the category of block ciphers. Although widely adopted and recognized as the standard symmetric key algorithm, DES is obsolete due to security concerns and has been replaced by newer algorithms. DES operates on 64-bit blocks, meaning it encrypts and processes 64 bits of plaintext to produce 64 bits of ciphertext. This encryption process involves the use of a 56-bit internal key or subkey [15]. Data security has great significance in the contemporary digital landscape, and the DES algorithm stands out as a widely used cryptographic algorithm to maintain data confidentiality and integrity. As a fundamental tool in the field of cryptography, DES plays a vital role in addressing the challenges posed by the ever-evolving digital era [16]. Plain text refers to the initial message or the original message sent in the communication process. communication process. This plain text undergoes the process of encryption and decryption. Cipher text, on the other hand, represents the hidden message resulting from the encryption of the original message (plain text) during the cryptography process. Cipher text can be converted back to its original form (plain text) using the provided key. The process involved in converting data (plain text) into disguised data (cipher text) is known as encryption. The DES algorithm is an encryption method used in block cipher systems. This encryption system scrambles data block by block, using 64-bit input blocks (plain text) and produces output (cipher text) in 64-bit blocks as well. The algorithm used is a symmetric key algorithm with a key length of 56 bits.

Originally established as a standard for securing transmitted and stored data, the DES cipher system quickly gained international adoption quickly gained international adoption for a variety of applications that required encryption during operation [17]. DES is a symmetric key block cipher characterized by a key length of 56 bits and a block size of 64 bits. Originated by IBM in 1972, DES was originally designed as a data encryption algorithm and was later adopted by the United States government as the standard encryption algorithm. Encryption algorithm originally implemented with 64-bit keys, the national security agency (NSA) later imposed a restriction, the key length to 56 bits. This resulted in DES discarding 8 bits from the original 64-bit key and using a compressed 56-bit key to encrypt data in 64-bit blocks. Despite its widespread adoption, DES has vulnerabilities, especially when weak keys are used. Various modes of operation, such as CBC, ECB, CFB, and OFB, increase its flexibility. In particular,DES faced a significant challenge in 1998 when the DES Cracker supercomputer, aided by many PCs distributed across the Internet, managed to crack DES in just 22 hours [18]. DES belongs to the Feistel cipher family, and its structure aligns with the typical characteristics of Feistel ciphers. However, there are specific details that distinguish DES within this framework: i) block length, the DES block length is 64 bits. Both the original plain text and the cipher text are 64 bits in size; ii) key size, the DES key is 64 bits in size. Each round of the DES algorithm uses a 48-bit round key; and iii) number of rounds, DES operates through 16 rounds. This means that the encryption or decryption process involves iterating through 16 consecutive rounds, each using a different integer key. These specifications contribute to the unique structure and functionality of DES as a Feistel cipher [19] shown in Figure 6.

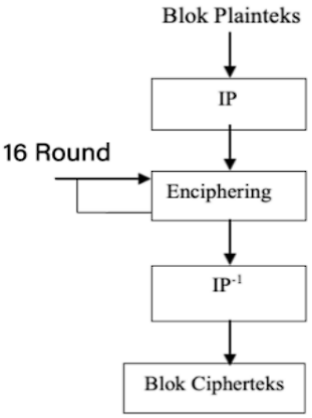

Figure 6. Global scheme of the DES algorithm

In the DES encryption process, the plain text block is initially divided into two part, the left part (L) and the right part (R). In the DES encryption process, the plain text block is initially divided into two parts: the left part (L) and the right part (R), each consisting of 32 bits. These two parts then undergo a series of 16 rounds in the DES encryption process. In each round, denoted as round i, the right block (R) serves as input to a transformation function known as f. In the function f, the right block (R) is combined with an internal key. This internal key, which is specific to each round, is derived from the master encryption key used in DES. The transformation function f plays an important role in transforming and mixing the bits in the right block, which contributes to the overall encryption process [15] because of the vulnerability and era of DES which is quite resistant, this encryption is added with BBS arithmetic to increase the randomness of the numbers produced.

2.3.3. Blum-blum-shub (BBS)

Algorithm BBS serves as a pseudorandom bit generator, producing a binary sequence known as the BBS sequence [19]. The process begins by choosing two prime numbers, p and q, both satisfying the condition p, $q \equiv 3 \pmod 4$ to ensure cryptographic security. The modulus n is then calculated as the product of these prime numbers ($n=p \times q$). Next, a random integer r is chosen such that it is prime to n, meaning it has no common factors with n. The initial seed is set as $x_0=r^2 \bmod n$, which serves as the starting point for generating the sequence. The pseudorandom sequence is then generated iteratively using the recurrence relation $x_{n+1}=(x_n)^2 \bmod n$, to ensure unpredictable and strong randomness properties. Each decimal digit I is converted to binary, and the least significant bit (LSB) is extracted to convert the generated random bits into binary form. By convention, 1 is represented as 1 and 0 as -1 in the representation. In a number of applications, the resulting 1 and -1 sequences can be used as spreading sequences [20] as shown in Figure 7.

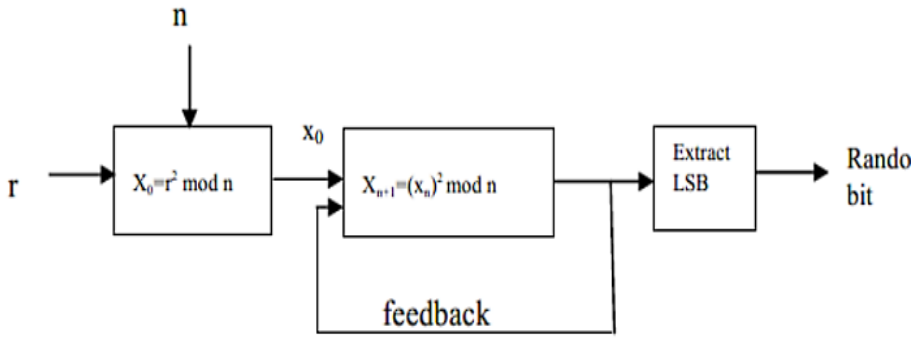

Figure 7. Block diagram for BBS sequence generator

The BBS number generator is one of the simplest and most powerful of the cryptographically secure pseudorandom number generators (CSPRNGs), particularly in terms of theoretical complexity. Developed in 1986 by Lenore Blum, Manuel Blum, and Michael Shub enerator BBS [12], the BBS algorithm uses the following equation to generate pseudorandom numbers. In this equation, Xn represents the current value in the sequence, and the mod n operation denotes the modulus operation with 'n'. The BBS algorithm, although simple in its formulation, exhibits strong cryptographic properties that make it suitable for secure pseudorandom number generation [21].

$$Xn + 1 = (xn)2mod\ n.$$

2.4. Research paper

To strengthen the foundation of this study, several related works on Botnet detection and prediction were reviewed. Previous research has employed a variety of approaches, ranging from recurrent neural networks (RNNs) for analyzing traffic patterns, to decision tree classifiers and social communication detection methods based on network flow. While these studies achieved promising results, many still face limitations such as high false positive rates, low recall values, or insufficient adaptability to evolving attack strategies. In contrast, the present study introduces a novel approach that integrates DES and BBS encryption techniques with a GRU-based deep learning model. This integration not only enhances prediction accuracy but also ensures the confidentiality of sensitive data, addressing a critical gap in prior research. A comparative summary of the methods, features, and performance metrics from earlier studies alongside the proposed system is presented in Table 2.

Table 2. Research paper

Research paper	Method	Features	Performance metrics (%)
[21]	Recurrent neural network	Size, duration, and periodicity	Accuracy: 97 False positive rate: 3.72
[11]	Social communication detection	Network flow-based	Recall: 2.6 Precision: 80 F1-score: 8.8
[14]	Decision tree	Network flow-based	Accuracy: 93 False positive rate:30 F1-score: 9.0
[9]	Periodicity in network flow	Periodicity in pcap data	F1-score: 9
[13]	Pattern-based network flow feature extraction	Statistical network flow-based	Accuracy: 99.36 Precision: 98.98 Recall: 98.47 F1-score: 98.72

3. METHOD

Botnet is one type of malware that often becomes a serious threat, especially for large companies and government agencies. This malware infects and controls a number of devices connected in a network, forming a network of IP bots that can be exploited by attackers. network, forming a network of IP bots that can be exploited by attackers without the knowledge of the device owner. Botnet attacks can be fatal, damaging network infrastructure, stealing sensitive data, and paralyzing critical services. Several previous

studies have developed prediction systems to detect and prevent Botnet attacks. This study focuses on developing such a prediction system by integrating data security techniques, namely encryption. DES and BBS encryption are implemented as an alternative to protect text data and prevent public data leaks, which are rampant in Indonesia. The prediction system model with encryption was developed by following the research flow illustrated in Figure 3. The preprocessing flow can be illustrated in Figure 8.

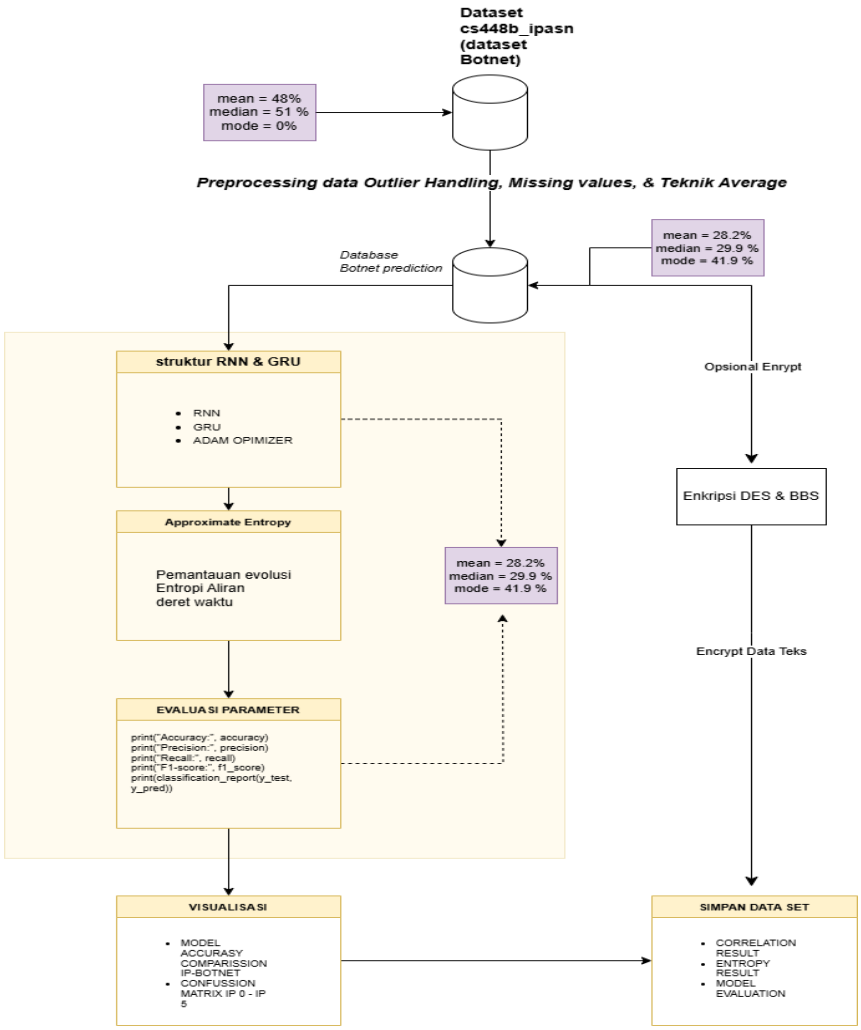

Figure 8. Data preprocessing flow

3.1. Research design

The developed model is a prototype of an information system implemented using Google Collaboratory and the Python programming language. The use of the Tesla T4 GPU on the Google Collaboratory platform aims to accelerate data processing, especially at the training and testing stages of models with large datasets. This allows for time efficiency and increased system performance.

3.2. Data collection and preparation

The data collection process begins by taking a dataset from automated-malware-analysis GitHub, specifically the cs448b_ipasn dataset. After that, data preprocessing is performed to clean and prepare the data for model training. This preprocessing includes handling outliers, dealing with missing values, applying averaging techniques, and encryption. Handling outliers on numeric data is done using the outliers_to_null_iqr(feature) function, which processes the DataFrame (series) column as input. The first step involves calculating the first quartile (Q1) and third quartile (Q3), where Q1 represents the value that separates the smallest 25% of the data, while Q3 separates the smallest 75%. The interquartile range (IQR) is then calculated as the difference between Q3 and Q1. IQR is used to determine the lower and upper limits for

outlier detection. The lower limit is determined by subtracting 1.5 times the IQR from Q1, while the upper limit is calculated by adding 1.5 times the IQR to Q3. Any value that falls outside these limits is considered an outlier and is replaced with np.nan (missing value). This process is applied iteratively to each numeric column in the DataFrame using a loop mechanism. The outliers_to_null_iqr() function is executed for each numeric column, with the results stored back into the corresponding column in the DataFrame. The output of this process is presented in Table 3.

Table 3. Sample data

Index	Date	I_ipn	r_asn	f	yday	wday
0	2006-07-01 0:00:00	0	436704.0	106.0	182	5
1	2006-07-01 0:00:00	1	182194.0	640.0	182	5
2	2006-07-01 0:00:00	2	212966.0	1677.0	182	5
3	2006-07-01 0:00:00	3	96376.0	22.0	182	5
4	2006-07-01 0:00:00	4	120507.0	184.0	182	5
5	2006-07-01 0:00:00	5	125647.0	44.0	182	5
6	2006-07-01 0:00:00	6	117227.0	63.0	182	5
7	2006-07-01 0:00:00	7	479524.0	411.0	182	5
8	2006-07-01 0:00:00	8	339609.0	448.0	182	5

The initial interpretation of each column in the dataset is as follows. Index represents a sequence number or unique identifier for each row of data. The date column records the date and time when the traffic data was collected. The ipn field refers to the "local IP network" or "local IP number," which indicates the local IP address or network. The rasn column stands for "remote autonomous system number (ASN)", which identifies the network from which the traffic originated. ASN represents a collection of networks under the control of a single entity, usually an internet service provider (ISP). The F column shows the number of flows or packets in the network traffic. The yday field represents the day of the year (e.g., January 1=1, December 31=365 or 366 in leap years), while the wday field indicates the day of the week (e.g., Monday=1, Sunday=7). Handling missing values in the dataset involves deleting the affected rows using the dropna() function with the inplace=true parameter. This ensures that the deletion is performed directly on the DataFrame without creating a new copy. Data visualization was then performed to analyze the trends by displaying ten line graphs, each depicting the number of flows (F) from ten different local IP addresses (ip0 to ip9) over time (yday). These graphs were created using matplotlib.pyplot, arranging the figures and axes in five rows and two columns. A horizontal line was added to each graph, representing the mean of F plus three times its standard deviation, serving as an upper bound to detect potential anomalies or outliers in the flow data. The purpose of this visualization was to observe trends and patterns in the number of flows from each local IP address over time, and the moving averages used to facilitate better visual reading in building models, identifying anomalies by comparing data points to the mean plus three standard deviations, and comparing flow patterns among different local IP addresses. The results of this visualization are presented in Figure 9.

Figure 9 shows a green horizontal line on each ip graph indicating a predetermined threshold or upper limit indicates a predetermined threshold or upper limit. Data above this line can be considered an anomaly or outlier. If seen in Figure 10 which displays a comparison of network traffic patterns between two local IP addresses, namely Local IP 1 to Local IP 9. traffic patterns between two local IP addresses, in Figure 10 the X-axis of both graphs shows the day of the year (yday), while the Y-axis shows the number of flows (possibly the number of data packets or connections) recorded on each IP address.

Interpretation of Figure 9 Local IP 1 shows a relatively stable traffic pattern between days 180 and 210, with little fluctuation. stable between days 180 and 210, with little fluctuation. There is a significant spike traffic spike around day 220, followed by a sharp decline until day 240. Local IP 4 shows a more unstable traffic pattern with many spikes and sharp declines throughout the observation period. There are several clearly visible traffic spikes, especially around yday 200, 225, and 270. Comparison of these two graphs shows that Local IP 4 has more volatile and unstable traffic than Local IP 1. Traffic spikes on Local IP 4 can indicate unusual activity, which exceeds the normal traffic threshold such as anomalous traffic attacks or Botnet activity, or significant changes in network usage by applying moving average techniques to smooth the data and visualize network traffic data trends. visualize network traffic data trends. which is used to reduce noise or random fluctuations in data, so that the general trend or pattern of the data is more easily seen. the general pattern of the data is more easily seen. Detecting long-term trends Identifying seasonal patterns and making it easier to predict future traffic.

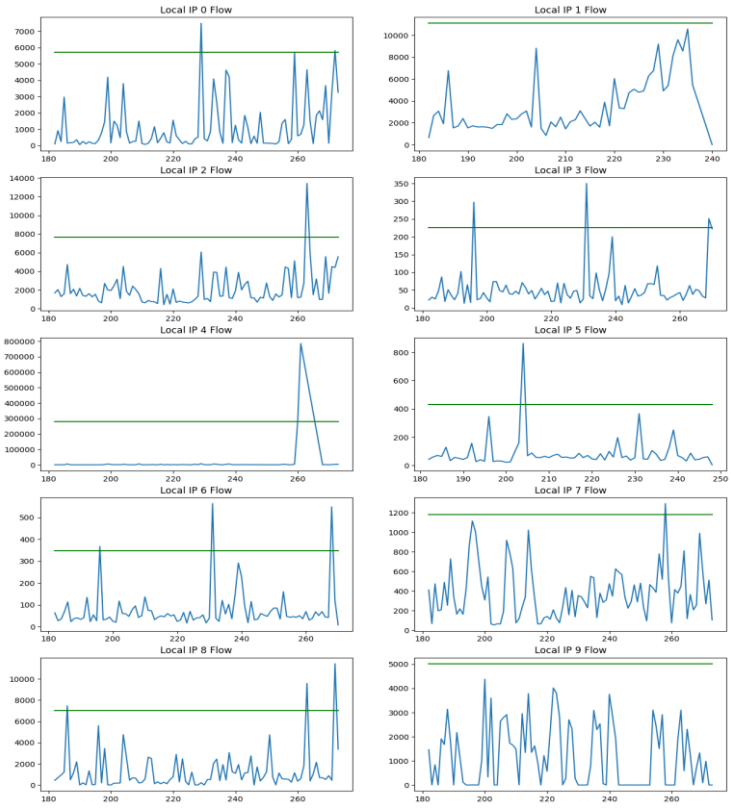

Figure 9. 10 traffic capture

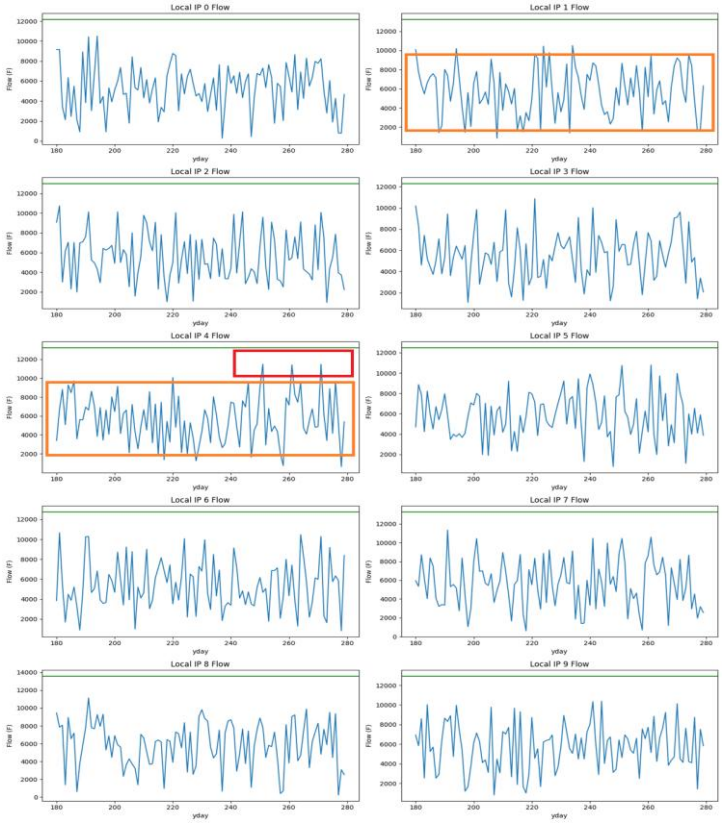

Figure 10. Comparison of analyzed traffic from Local IP 9-Local IP 9

3.3. Botnet prediction model training

The processed dataset is then used to train the Botnet prediction model. The results of the preprocessing above produce a significantly changed in Figure 11. Significant that we can see from the bar chart of Figure 12 which is the after and before of the traffic that has been applied to the machine learning method.

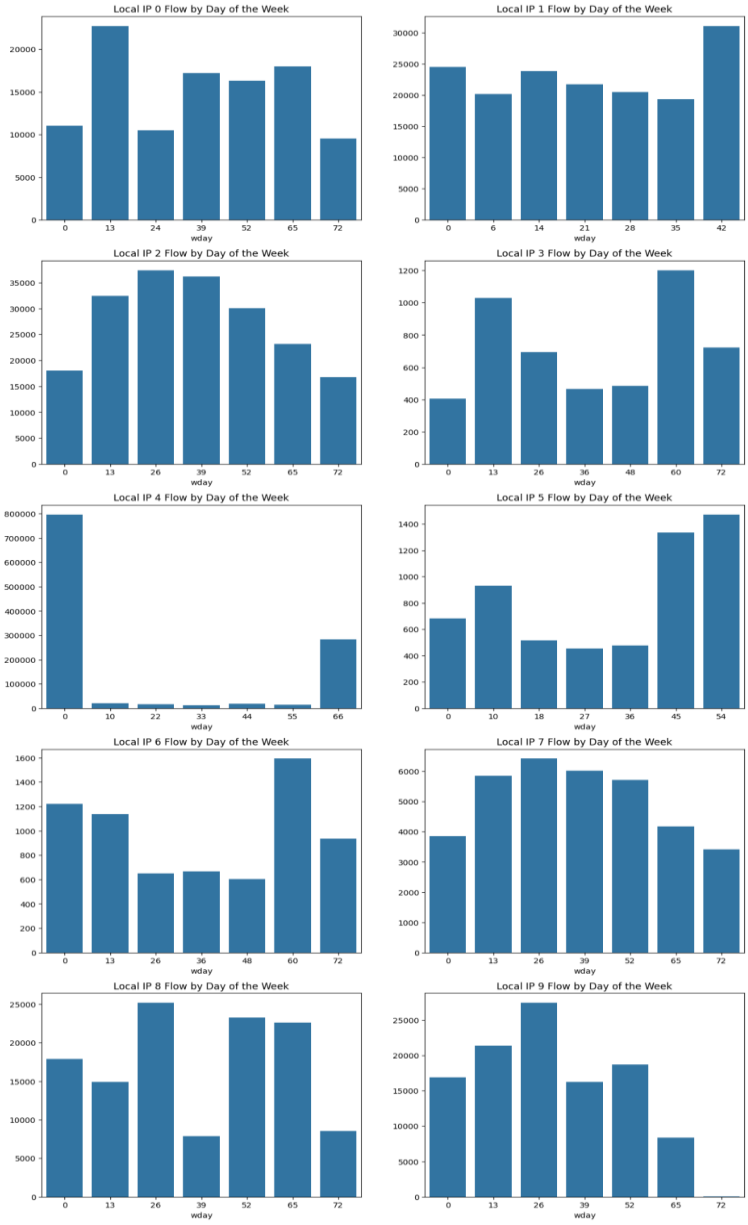

Figure 11. Before preprocessing using outliers and missing values

Evaluation of the preprocessing results revealed several positive impacts. The dominance of outliers was significantly reduced, as indicated by the smaller Y-axis scale and more stable graph patterns. In addition, the network traffic patterns appeared more consistent and easier to interpret after preprocessing. Once the preprocessing process was completed, the next step involved building the model. The chosen model used the RNN (GRU) structure, taking advantage of the capabilities of these algorithms in handling sequential data, such as time series data on network traffic. Before feeding the data into the model, it was scaled to a range of 0 to 1. This scaling process has two main purposes: ensuring that all values are in the same range and accelerating the convergence of the machine learning model, especially for algorithms that are sensitive to data scale, such as gradient descent. To properly structure the time series data for the RNN

model, the create_dataset function was used. This function reformats the data to account for dependencies across different time steps. The create_dataset function includes key parameters: dataset, which represents the time series data in the form of a Pandas DataFrame or Series, and look_back, which specifies the number of previous time steps used as input to predict the next time step.

Model optimization is performed using the Adam algorithm and early stopping. The implementation of this optimization involves several steps. First, the required libraries are imported. The GRU, dense, and dropout modules from TensorFlow Keras are used to build the model, while Adam is used as the optimizer. In addition, early stopping is implemented to prevent overfitting in data processing in machine learning, and numpy is used for array operations. The look_back parameter is set to 14, which specifies the number of previous time steps used as input to predict the next time step. The train model (data) function is then defined to accept the time series data as input. In the preprocessing step, the dataset is converted to a NumPy array using data.values. A list of target values (Y) and an input sequence (X) are created, where X consists of the sequence of data points along the look_back, and Y represents the anticipated value. The data is then reshaped into NumPy arrays (trainX and trainY), ensuring that trainX matches the expected input shape for a GRU model, with dimensions [number of samples, time steps, number of features]. Next, the model is built using a sequential architecture. A GRU layer with 128 units is added first, with return_sequences=true to allow other GRU layers to follow. A Dropout layer with a rate of 0.2 is included to prevent overfitting. A second GRU layer with 64 units is then added, followed by a Dense layer with one output unit to make predictions. To compile the model, the Adam optimizer was applied with a learning rate of 0.001 using optimizer=Adam (learning_rate=0.001). The model was compiled using the model.compile() function, combining the Adam optimizer and mean_squared_error as the loss function. During model training, an EarlyStopping callback was defined to stop training if the val_loss did not improve after 10 epochs, thus preventing overfitting. The model.fit() function was used to train the model with the trainX and trainY datasets. Training parameters included a maximum of 500 epochs, a batch size of 32, and verbose=0 to suppress training output. Additionally, the EarlyStopping callback was integrated with callbacks=[early_stopping]. The data was split into 90% training and 10% validation using validation_split=0.1. After training was complete, the trained GRU model was returned. Finally, the trained model was used to generate predictions for the time series data using the predictFlow function. This function takes a data pattern as input, uses a trained model to make a prediction, and then returns the result, as shown in Figure 13 from the output.

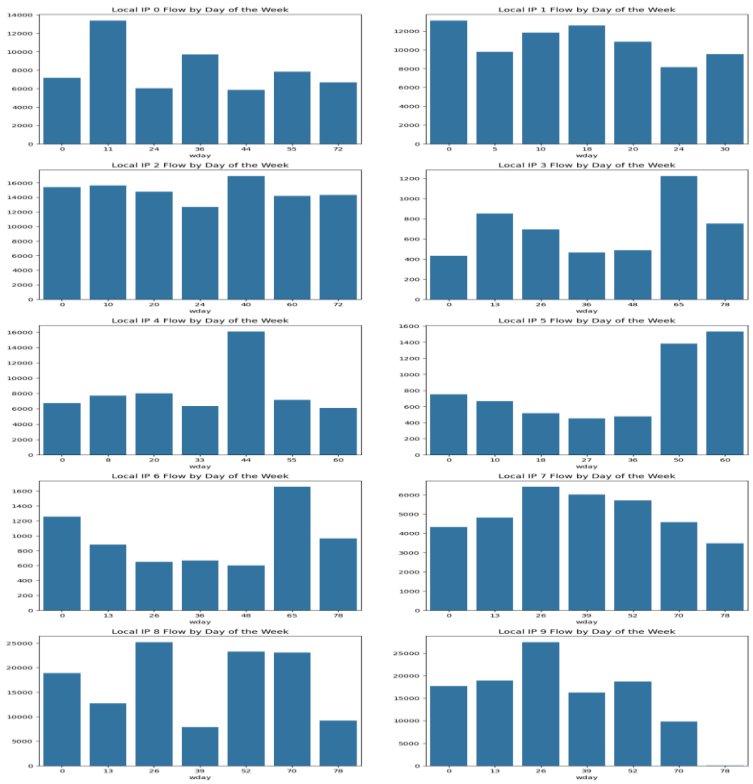

Figure 12. After preprocessing using outliers and missing values

Figure 13. The result of predicting the number of network traffic flows

The model demonstrates strong performance in predicting network traffic trends and patterns across most IP addresses. In several cases, such as Local IP 0, Local IP 3, and Local IP 6, there is a close alignment between the red line (predicted) and the blue line (actual), indicating accurate forecasting of traffic behavior. However, the model shows less precision in capturing sudden traffic spikes for certain addresses, including Local IP 1, Local IP 4, and Local IP 9. To provide a comprehensive evaluation of these results, the performance metrics are summarized in Table 4, which highlights the accuracy, precision, recall, and F1-score achieved by the GRU model.

Table 4. Conffusion matrix

Conffusion matriks GRU
[[168]]
Accuracy (GRU): 100.00%
Precision (GRU): 1.00
Recall (GRU): 1.00
F1-score (GRU): 1.00

3.4. Network traffic analysis and Botnet detection

A thorough analysis of the network traffic patterns reveals significant differences between different IP addresses. For example, there is a substantial and fluctuating traffic spike on Local IP 4 compared to other IP addresses. This fluctuation indicates a variation in the activity pattern that may signal a change or anomaly in the network traffic. To ensure the quality and reliability of the data used in model training, data validation and entropy analysis are performed. Data validation is conducted periodically to maintain consistency and accuracy throughout the training process, while entropy analysis is applied to measure the complexity and randomness of the data. This analysis helps in understanding the characteristics of the dataset and identifying

unusual patterns. Furthermore, the temporal evolution of network traffic is monitored to observe changes in traffic behavior over time, allowing the identification of trends and anomalies that can support early detection of potential problems or threats. The evaluation results of modeling and prediction accuracy are summarized in Table 5, which presents detailed performance metrics for each IP address.

Table 5. Results of modeling and prediction of network accuracy indicated as BOT

IP	Accuracy	Precision	Recall	F1	Cm
b'\x83\x15A\x1d\xe1Y\x9e\xae'	b"\xfc\xb1\xf1\xc43U'\x0f"	b'X\x94\xb9\xc8\xaa!`\xcd8\xb3|\xa4\x81{a3{\xdf\x1a\xbbj\xec\x1dB'	b"\xfc\xb1\xf1\xc43U'\x0f"	b'F\xff\xac\xd6 \xd5G\x9c\xda\xfbMa\x05z\xd5\x91\xa2\xdb\x92\x1cD\xca\x8ab'	b'\xa5\xc8Q&\xa7\xda\xc2c\x99[\xb4\x81j$\xf8\x966\xa1 \xb9\xba%\xb3Q'
b'\x85=u\xc8=\xf1}J'	b'\xf8\xc3\xfbI7?\xfd\x8d'	b'\x1d\x1f\xac\t\x1f\x06\x17\x18ZZ\x8b\xe3\x1bc\xb1\x1f\xc0b\xdaaA\xe4\xa4\xd9'	b'\xf8\xc3\xfbI7?\xfd\x8d'	b'\xf8\xc3\xfbI7?\xfd\x8d'	b'\x9e\x1d\x11\xc5\x9f\x11"\x9a\xfc\xdf+\x90\xb0\x05\xa5S6\xa1 \xb9\xba%\xb3Q'
b'\x85#\xc9c\xa7\x1b\xbdT'	b'\x98\xc74\xbb\x94n!#'	b'\xe1\x94\x08\xbe\xc8e~\xd9'	b'\x98\xc74\xbb\x94n!#'	b'4\x97\xbdR\xa6~#\x90\xe5u\xc41\xc3\xf5LD\xb5Y\x19&\x90b\x11Z'	b'T\xcc\x99"z0W\xab\x9e\x1fA\x17!\x1eK16\xa1 \xb9\xba%\xb3Q'
b"\x10\x84\x1c'6pX7"	b'x\x97\xde\xcd4\xb8\xcd\x9b'	b'\xcc\x04uw.\xbe8\x84R\xcd\xc0\xbd\x88;z\x86\xa3\xef\xbc\xafiK\xf9\x88'	b'x\x97\xde\xcd4\xb8\xcd\x9b'	b'`4\x06\xa9e\xf3j\xaa\x8a\x1by\xcd\x90O\x01\x93U\x89h\x11\xad\xd5\xe9'	b'\xa5\x1cE\\{N\t\xe0\xa4\xee2oBu-"6\xa1 \xb9\xba%\xb3Q'
b'I\n\xae\xeeI>\xa8\xd3'	b'^\xd6/\xf2\xee\xcaes'	b'\xd4\x1b\x99\xb2\x11B\xd1r\xc7\\\xa4g\x8c\x9a\x04\xfdo0\xf2\xe1\xa1\x0c\xd0w'	b'^\xd6/\xf2\xee\xcaes'	b'\x0ca5\x8ev71O\x90Y\xcf\x08\xf5\xc8\xc1\xb8x-\'"\xbb\x1eoT'	b'\xf8\xbf\x8c\x95{x\x00\x1c\x7f\xac\xb1\xe3\xc1\xb1\xb8\xb46\xa1 \xb9\xba%\xb3Q'
b'\x89j\xa2pm\x8b\xbb\xfe'	b'\xc1\xac\x1e\xae\xa6\xc2\xe98'	b'o\x99\xec\x11\x86\x88\\I\x03\xe5mN\xa2\xa2|\xbf\xe4E.3F\xe9\xc7\xda'	b'\xc1\xac\x1e\xae\xa6\xc2\xe98'	b'^I0*\x0cSK\xa44\x14\xc3z\x04\xbdLZ,4\xc5\xdc\xc7\xd8\xc5\xe5'	b'i\xaa2\x8e\xeeZ\x8a\xab\xbb6\x103\xc1^\x9e\xf36\xa1 \xb9\xba%\xb3Q'

This encryption result can be decrypted with a simple result example. b'\x83\x15A\x1d\xe1Y\x9e\xae becomes ip0 b"\xfc\xb1\xf1\xc43U'\x0f" becomes 0.51 accuracy value, b'X\x94\xb9\xc8\xaa!`\xcd8\xb3|\xa4\x81{a3{\xdf\x1a\xbbj\xec\x1dB' becomes 0.5105929487179488 precision, b"\xfc\xb1\xf1\xc43U'\x0f" becomes 0.51 recall value, b'F\xff\xac\xd6 \xd5G\x9c\xda\xfbMa\x05z\xd5\x91\xa2\xdb\x92\x1cD\xca\x8ab' becomes 0.5102456140350877 value f1 score, and b'\xa5\xc8Q&\xa7\xda\xc2c\x99[\xb4\x81j$\xf8\x966\xa1 \xb9\xba%\xb3Q' becomes the correlation matrix (CM) value to [[28 25] [24 23]] or in Table 6.

To further evaluate the performance of the proposed GRU-based prediction system, a detailed analysis of model accuracy, precision, recall, and F1-score was conducted across multiple IP addresses. This evaluation highlights how the model performs under varying traffic conditions and provides insight into its ability to distinguish between normal and anomalous patterns. The results are summarized in Table 6, which presents the performance metrics for each IP, including confusion matrix values that reflect classification reliability.

Table 6. Model evaluation matriks

IP	Accuracy	Precision	Recall	F1	Cm
IP0	0.51	5.105.929.487.179.480	0.51	5.102.456.140.350.870	[[28 25] [24 23]]
OP1	0.46	4.603.681.472.589.030	0.46	0.46	[[23 28] [26 23]]
IP2	0.49	49.125	0.49	48.821.052.631.578.900	[[27 22] [29 22]]
IP3	0.45	44.915.458.937.198.000	0.45	449.172.932.330.827	[[25 26] [29 20]]
IP4	0.55	5.740.147.783.251.230	0.55	5.493.249.324.932.490	[[28 15] [30 27]]
IP5	0.47	4.756.896.551.724.130	0.47	467.077.694.235.589	[[26 21] [32 21]]

From these results, any data with accuracy exceeding 51% can be considered indicative of normal traffic behavior. To extend the evaluation, a 14-day prediction horizon was applied to assess the model's ability to forecast traffic patterns over time. The outcomes of this longer-term prediction are presented in

Table 7, which demonstrates the accuracy and precision achieved for each local IP address during the extended testing period.

Table 7. 14-day prediction evaluation

Local IP	Prediction horizon (14 days)	Accuracy	Precision
0	14	0	0
1	14	7.14%	100
2	14	0	100
3	14	85.71%	100
4	14	0	100
5	14	85.71%	100
6	14	85.71%	100
7	14	7.14%	100
8	14	0	100
9	14	42.86%	100

4. RESULTS AND DISCUSSION

The evaluation of the proposed GRU-based prediction system begins with an examination of the time series forecasting results. By analyzing the predicted versus actual traffic flows, the model's ability to capture network behavior and anomalies can be assessed. As illustrated in Figure 13, the time series predictions provide valuable insights into the alignment between expected and observed traffic patterns, highlighting both the strengths and limitations of the model in handling dynamic network environments. Following this, additional analyses such as approximate entropy (ApEn) variation (Figure 14) are employed to measure irregularity and complexity in the data series. These complementary evaluations, combined with preprocessing and optimization techniques, enable more precise detection of suspicious traffic patterns, including botnet activity. The integration of encryption further ensures data confidentiality, preventing leakage and reinforcing the robustness of the system against vulnerabilities.

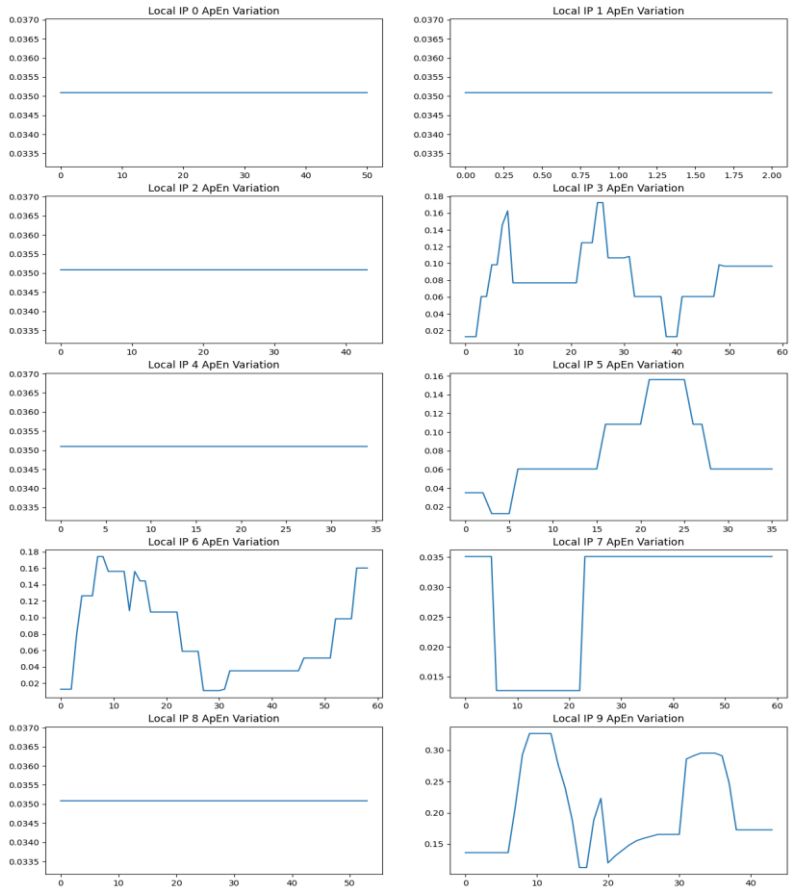

Figure 14. Approximate entropy (ApEN) variation, to analyze the irregularity in a series of data

The ApEn values of IP flows across ten different IP addresses are shown to vary in the graph above. Time series with high ApEn values are complex and erratic, while those with low ApEn values are more predictable. The use of ApEn analysis in conjunction with data preprocessing methods has enabled us to generate more precise and focused prediction findings. Data processing is also made easier by the use of various optimization techniques, especially when dealing with suspicious traffic such as botnets or other attacks. As a result, network security can be improved and network performance degradation due to excessive traffic load can be avoided. To maintain data confidentiality, encryption is used as an additional security measure alongside traffic analysis. By using this method, data leakage from security breaches—which are often triggered by system vulnerabilities—can be prevented.

5. CONCLUSION

This study presents a promising framework for secure and accurate botnet host prediction by integrating deep learning with encryption techniques. The use of GRU-based modeling combined with DES and BBS encryption demonstrates that it is possible to achieve high predictive accuracy while simultaneously safeguarding sensitive data, thereby addressing both performance and security challenges in network protection. The results confirm that the proposed approach can effectively detect malicious hosts and enhance the reliability of intrusion detection systems. Future research should focus on expanding the dataset to include larger and more diverse traffic samples, exploring alternative or hybrid encryption algorithms to further strengthen data confidentiality, and investigating other advanced machine learning architectures to improve adaptability against evolving cyber threats. By continuing to refine and extend this approach, the system has the potential to contribute significantly to the development of resilient cybersecurity infrastructures, supporting both academic advancements and practical applications in protecting critical digital environments.

ACKNOWLEDGMENTS

Throughout this investigation, we would like to express our sincere gratitude for the essential advice, encouragement, and support provided by our team and colleagues. We also extend our appreciation to Universitas Nusa Mandiri for granting us the opportunity to conduct this research.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Omega Joel Patria Moata	✓	✓	✓	✓	✓	✓		✓	✓	✓			✓	
Irwansyah Saputra		✓				✓		✓	✓	✓	✓	✓		

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

DATA AVAILABILITY

The authors confirm that the data supporting the findings of this study are available within the article.

REFERENCES

[1] L. Oluwaseye, J. A1, W. Doorsamy, and S. Paul, "A review of missing data handling techniques for machine learning," *International Journal of Innovative Technology and Interdisciplinary Sciences*, vol. 5, no. 3, pp. 971–1005, 2022, doi: 10.15157/IJITIS.2022.5.3.971-1005.

[2] I. Febyola, Nurul Annisa, Arista Sidabutar, "The Role of Cybersecurity in Protecting Indonesian Citizens' Data (Case Study: Bjorka Hacker)," (in Indonesian), *Bidang Penelitian Informatika*, vol. 1, no. 1, 2022, [Online]. Available: https://ejournal.kreatifcemerlang.id/index.php/jbpi/article/view/78/8

[3] D. Septasari, "The security and the challenge of society 5.0 era in Indonesia," *Aisyah Journal Of Informatics and Electrical Engineering*, vol. 5, no. 2, pp. 227–233, 2023, doi: 10.30604/jti.v5i2.231.

[4] D. O. Wijoyo, "Social Media Analysis of the Issue of Indonesia as the Most Impolite Country in Southeast Asia," (in Indonesian), *Jurnal Riset Manajemen Komunikasi*, vol. 3, no. 1, pp. 1–6, 2023, doi: 10.29313/jrmk.v3i1.1015.
[5] A. Ariska and W. Wahyuddin, "Application of Cryptography Using the Data Encryption Standard (DES) Algorithm," (in Indonesian), *Jurnal Sintaks Logika*, vol. 2, no. 2, pp. 9–19, 2022, doi: 10.31850/jsilog.v2i2.1734.
[6] S. H. Waruwu and R. K. Hondro, "Analysis and Implementation of a Modified GOST Cryptographic Algorithm Using the Blum Blum Shub Generator for Website Login Security Systems," (In Bahasa), *KETIK : Jurnal Informatika*, vol. 1, no. 3, pp. 30–46, 2024.
[7] E. Gomede, R. M. de Barros, and L. de S. Mendes, "Deep auto encoders to adaptive e-learning recommender system," *Computers and Education: Artificial Intelligence*, vol. 2, 2021, doi: 10.1016/j.caeai.2021.100009.
[8] M. Schuld, R. Sweke, and J. J. Meyer, "The effect of data encoding on the expressive power of variational quantum machine learning models", doi: 10.48550/arXiv.2008.08605.
[9] R. S. S. Moorthy and N. Nathiya, "BotNet detection using artificial intelligence," *Procedia Computer Science*, vol. 218, pp. 1405–1413, 2023, doi: 10.1002/9781119760429.ch4.
[10] A. Encalada-Davila, L. Moyon, C. Tutiven, B. Puruncajas, and Y. Vidal, "Early fault detection in the main bearing of wind turbines based on gated recurrent unit (GRU) neural networks and SCADA data," *IEEE/ASME Transactions on Mechatronics*, vol. 27, no. 6, pp. 5583–5593, 2022, doi: 10.1109/TMECH.2022.3185675.
[11] S. García, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *Computers and Security*, vol. 45, pp. 100–123, 2014, doi: 10.1016/j.cose.2014.05.011.
[12] R. Yacouby and D. Axman, "Probabilistic extension of precision, recall, and F1 score for more thorough evaluation of classification models," in *Proceedings of the First Workshop on Evaluation and Comparison of NLP Systems*, 2020, pp. 79–91. doi: 10.18653/v1/2020.eval4nlp-1.9.
[13] A. Rehman Javed, Z. Jalil, S. Atif Moqurrab, S. Abbas, and X. Liu, "Ensemble Adaboost classifier for accurate and fast detection of botnet attacks in connected vehicles," *Transactions on Emerging Telecommunications Technologies*, vol. 33, no. 10, 2022, doi: 10.1002/ett.4088.
[14] A. E. Maxwell, T. A. Warner, and F. Fang, "Implementation of machine-learning classification in remote sensing: an applied review," *International Journal of Remote Sensing*, vol. 39, no. 9, pp. 2784–2817, 2018, doi: 10.1080/01431161.2018.1433343.
[15] L. Göcs and Z. Csaba JOHANYÁK, "Catboost algorithm based classifier module for brute force attack detection," *International Journal of Engineering*, pp. 13–18, 2023.
[16] A. Vuppala, R. S. Roshan, S. Nawaz, and J. V. R. Ravindra, "An efficient optimization and secured triple data encryption standard using enhanced key scheduling algorithm," *Procedia Computer Science*, vol. 171, pp. 1054–1063, 2020, doi: 10.1016/j.procs.2020.04.113.
[17] M. Alkaff, A. Rizky Baskara, and Y. Hendro Wicaksono, "Sentiment analysis of Indonesian movie trailer on YouTube using delta TF-IDF and SVM," *2020 5th International Conference on Informatics and Computing, ICIC 2020*, 2020, doi: 10.1109/ICIC50835.2020.9288579.
[18] D. Wang and H. Qian, "CatBoost-based automatic classification study of river network," *ISPRS International Journal of Geo-Information*, vol. 12, no. 10, 2023, doi: 10.3390/ijgi12100416.
[19] E. O. Ogunseye, C. A. Adenusi, A. C. Nwanakwaugwu, S. A. Ajagbe, and S. O. Akinola, "Predictive analysis of mental health conditions using AdaBoost algorithm," *ParadigmPlus*, vol. 3, no. 2, pp. 11–26, 2022, doi: 10.55969/paradigmplus.v3n2a2.
[20] T. Estella, N. Andrita Intan Ghayatrie, and A. Wibowo, "Outlier handling in clustering: a comparative experiment of kmeans, robust trimmed k-means, and k-means least trimmed squared," *International Journal of Computing and Digital Systems*, vol. 16, no. 1, pp. 1029–1039, 2024, doi: 10.12785/ijcds/160175.
[21] Z. Zainuddin, E. A. P. Akhir, and M. H. Hasan, "Predicting machine failure using recurrent neural network-gated recurrent unit (RNN-GRU) through time series data," *Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 2, pp. 870–878, 2021, doi: 10.11591/eei.v10i2.2036.

BIOGRAPHIES OF AUTHORS

Omega Joel Patria Moata holds a Diploma in Computer Technology from Universitas Bina Sarana Informatika and a Bachelor's degree in Informatics Engineering from Universitas Nusa Mandiri, where he self-financed his studies and graduated in 2021. He has been working as a network operation and technology professional at PT. IFORTE SOLUSI INFOTEK since 2020. He holds international certifications as a CCNA CyberOps associate and ethical hacker from Cisco Academy. His professional interests include network security, cybersecurity operations, and IT infrastructure. He can be contacted at email: 14230016@nusamandiri.ac.id.

Irwansyah Saputra holds a Bachelor's degree in Informatics Engineering from Universitas Nasional PASIM (2015) and a Master's degree in Computer Science from Universitas Nusa Mandiri (2019). He later completed his doctoral studies at the Graduate School of FMIPA IPB in 2024 through self-financing. Since 2019, he has been a lecturer at Universitas Nusa Mandiri in Jakarta. In addition to teaching, he is actively involved in research projects, technological development, and training in the fields of data science and machine learning. He holds an international certification as a Microsoft Technology Associate for Database Fundamentals. He can be contacted at email: Irwansyah.iys@nusamandiri.ac.id.