Smart Contract-Enhanced Residual GRU with Merkle-Damgård Cryptography for IoT Attack Detection

T. Nishitha
Gokaraju Rangaraju Institute of Engineering and Technology, Hyderabad, India
tnishitha1217@gmail.com (corresponding author)

Akhil Khare
Department of CSE, Maturi Venkata Subba Rao Engineering College, Hyderabad, India
khare_cse@mvsrec.edu.in

Received: 30 August 2024 | Revised: 18 September 2024| Accepted: 15 November 2024

 DOI: https://doi.org/10.48084/etasr.8860

ABSTRACT

As IoT continues to expand, the security of connected devices remains a critical concern, particularly in the face of DDoS attacks. This study introduces a novel approach that leverages blockchain technology through smart contracts integrated with an advanced attack detection mechanism. Central to this approach is the Enhanced Residual Gated Recurrent Unit (ERGRU) architecture, designed to effectively identify and mitigate DDoS attacks within IoT networks. The Adaptive Coati Optimization Algorithm (ACOA) was used to adjust the hyperparameters of the ERGRU model, such as the learning rate and the number of GRU neurons, to further improve detection accuracy. In addition, the proposed framework uses a one-way compression function to generate secure hashes for input data, utilizing the Merkle-Damgård cryptography technique to ensure data integrity and confidentiality. The proposed solution was tested through a rigorous process using a DDoS dataset. Performance was assessed by focusing on metrics such as processing time, data integrity rate, and confidentially rate. The results demonstrate the effectiveness of the proposed smart contract-based framework in providing a durable and efficient protection mechanism against DDoS attacks in IoT environments.

Keywords-IoT; distributed denial of service; residual gated recurrent unit; Coati optimization algorithm; Merkle-Damgård cryptographic technique

I. INTRODUCTION

An IoT network consists of items that perceive, interact, and communicate with the outside environment [1]. Everyday life involves several IoT applications, such as traffic control, e-health, smart cities, smart automobiles, and smart homes. The IoT is expected to reach more than 27 billion connections by 2025, increasing at a rapid pace [2]. Everything is online, and a plethora of new IoT gadgets emerge every day. IoT technologies have enabled a massive increase in the number of heterogeneous devices connected to the Internet. The growth of IoT has been one of the most significant technological advances in the last 10 years. The IoT fundamentally altered technology by allowing real objects to connect and communicate with each other online to improve human lives [3, 4]. The IoT epitomizes the transformation of the digital landscape, moving beyond traditional devices such as computers and smartphones to create an interconnected web of everyday objects, being the cornerstone of the 21st-century revolution [5, 6].

IoT devices are one of the main targets of threats. Recent reports claim that DDoS attacks and botnets account for most cyber-attacks that occur nowadays. The frequency and severity of these episodes have increased dramatically during the last ten years [7-10]. Traditional security measures, especially centralized intrusion detection systems, are not well equipped to handle the intricacies of IoT. These centralized systems often suffer from scalability issues and are struggling to monitor the massive data flows generated by the plethora of IoT devices. Moreover, centralized systems also introduce a single point of failure, making them attractive targets for adversaries [11-13].

Many secure data storage and attack detection systems have been proposed for IoT environments. In [14], Principal Component Analysis (PCA) was used along with the Random Forest (RF) classifier, achieving high precision, recall, F1-score, accuracy, and kappa coefficients. On the other hand, the Naive Bayes classifiers performed comparatively worse. In [15], behavior models based on Stochastic Petri Nets (SPNs) and an iterative computational technique with linear complexity in the number of nodes were introduced. This made it possible to specify and examine attack and defense plans for

performing voting-based IDS operations. In this method, good nodes could maximize the system lifetime. However, attackers target higher-capacity nodes more frequently than lower-capacity ones in an attempt to bring about an application failure.

In [16], a two-pronged machine-learning technique was proposed to detect and prevent IoT botnet attacks. The performance of the introduced ResNetScan-1 and ResNetDDoS-1 models was evaluated using scans and DDoS traffic samples from three publicly available datasets. The ResNet-18 model was trained over these datasets and the resulting ResNetScan and ResNetDDoS models were saved. However, the latter was unable to fairly identify DDoS traffic, having an average F1-score far below the typical precision. In [17], a tool known as C-DAD (Counter-based DDoS Attack Detection) was developed, which efficiently detected DDoS attacks using values of various network properties. The performance of this framework was analyzed in detail on several aspects such as SD-IoT network throughput, CPU and memory consumption, and attack detection time. However, security is one of the main issues with an IoT ad-hoc system.

In [18], Federated Learning (FL) was used to strengthen the security landscape of IoT networks, focusing specifically on the detection of DDoS attacks. The proposed FL-DAD method underscored the efficacy of decentralizing the learning process, ensuring data privacy while not compromising detection accuracy. The numerical results demonstrated that this approach achieved an accuracy consistently above 98% across various DDoS attack classes.

II. OPTIMIZING IOT ATTACK DETECTION WITH SMART CONTRACTS AND COMPRESSION TECHNIQUE

A. *Overview*

This study aims to improve the detection of IoT attacks through the application of blockchain and deep learning. Data are first collected using IoT sensors. A one-way compression function is integrated into the blockchain architecture to protect data confidentiality. This feature ensures that the information is kept private and safe. Subsequently, an attack detection mechanism using an Enhanced Recurrent Gated Recurrent Unit (ERGRU) model is added to the blockchain smart contract. This model is necessary to identify and evaluate abnormalities suspicious of assaults. The ERGRU's hyperparameters, such as learning rate and GRU neuron count, were set using an Adaptive Coati Optimization Algorithm (ACOA) to achieve optimal performance. All input data are hashed using the one-way compression function in the Merkle-Damgård cryptography technique, ensuring data integrity and confidentiality. Finally, the processed and hashed data is uploaded to the blockchain, providing the foundation for real-time attack detection in the IoT environment. Figure 1 illustrates the workflow.

B. *Enhanced RGRU Architecture for Smart Contract Detection*

The system collects data from IoT sensors and divides them into various categories to start the classification process. An enhanced RGRU architecture, optimized with the help of the ACOA, is used to achieve this. The data are labeled to differentiate between usual and potentially dangerous behavior. To correctly detect and prevent potential attackers in the IoT, the proposed blockchain-based smart contract must complete this classification process.

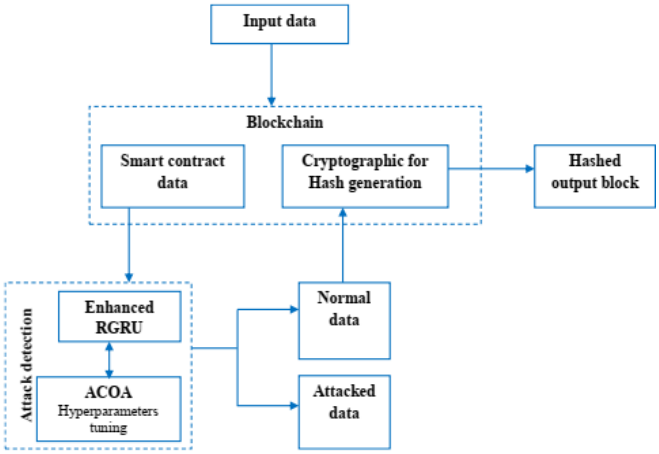

Fig. 1. Flow chart for the proposed structure.

1) *Enhanced RGRU Architecture*

This study leveraged an enhanced Long Short-Term Memory (LSTM) neural network, known as RGRU, to enhance the IoT vulnerability detection mechanism. The simpler structure of GRU, compared to LSTM, results in faster training and higher computing efficiency, making it ideal for this setting. Figure 2 shows the neural structure diagram of the deep-enhanced RGRU neural network, which is derived from the GRU neural network.

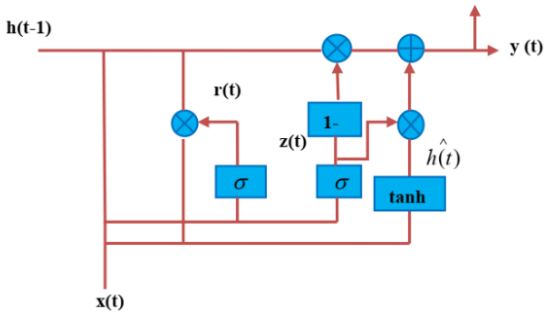

Fig. 2. Enhanced RGRU structure.

Figure 2 shows how the improved RGRU neuron varies from the GRU neuron because the reset gate can rescreen the current input data x_t, as in the update gate z_t, the r_t is multiplied with prior values to conceal the state weight. In other words, the output of the reset gate is utilized to alter the update gate to improve the neuron system, where the weight between the input is denoted as ω_z and the update gate is denoted as h_{t-1}. The reset gate's weight between input and h_{t-1} is represented by ω_r. $\omega_0$represents the weight of h_t. The symbols z_t and r_t in this formula indicate the same function as

regular GRU neurons among them. The series of inputs is assumed to be (x_1,x_2,..., x_t), and the formula for computing a typical enhanced RGRU output for a unit is described below.

Reset gate calculation:

$$r_t = \sigma(\omega_r * [h_{t-1}, x_t]) \quad (1)$$

Update Gate Calculation:

$$z_t = \sigma(\omega_z * [h_{t-1}, x_t * r_t]) \quad (2)$$

Candidate hidden state calculation:

$$\overline{h_t} = tanh\left(\omega * \left[r_{\ t} * h_{t-1}, x_t\right]\right) \quad (3)$$

Final hidden state calculation:

$$h_t = (1 - z_t) * h_{t-1} + z_t * n_t \quad (4)$$

$$y_t = \sigma(\omega_0 * h_t) \quad (5)$$

The ERGRU neural network, which reduces gradient attenuation and simplifies hidden state identification, has a more efficient neuron structure than the standard GRU, as shown in (2) and Figure 2. As a result, the RGRU model can retain long-term relationships while improving accuracy and learning efficiency. Three layers comprise the network: input, hidden, and output. The hidden layers of the network contain sophisticated RGRU neurons. Performance is enhanced overall by optimizing the ERGRU model, which also improves information retention and effective information propagation.

a) Residual Network

Due to the limited learning efficiency of GRU, the slow convergence rate, and the complexity of time series data, it is sometimes not possible to eliminate all duplicate state information in a single screening. Therefore, the improved RGRU neural network is used. Residual Networks (ResNets) take advantage of residual blocks, which, include skip connections, and a quick path for gradient flow. The network's capacity to transfer information is increased by adding gradients element by element through the use of residual blocks. ResNets are very useful for extracting complex and abstract data. Skip connections are required to prevent feature loss and maintain past data, helping the model learn and extract more dependable features. Two 1D convolutional layers with a filter size of one, batch normalization, and a Rectified Linear Unit (ReLU) activation function are usually used to construct each residual block. A ResNet has a convolutional weight layer, referred to as $f(x)$, where the input is x and the output is $h(x)$. The following equation describes the relationship between these elements:

$$h(x) = f(x) + x \quad (6)$$

In the residual network, the input x is directly connected to the network's output. Equation (2) illustrates that the network will learn its residual rather than the best mapping function directly at this stage.

$$f(x) = h(x) - x \quad (7)$$

The model continues to learn the residual mapping as the network expands, ensuring that the gradients do not vanish as the network deepens. If the residual mapping reaches zero ($f(x) = 0$), the output $h(x)$ equals the input x, theoretically keeping the network in an optimal state even as network depth increases. This study aimed to enhance the efficacy of the threat detection system in the blockchain-based IoT architecture using the efficiency and simplicity of RGRU. This approach ensures the fast and precise identification of security threats, thus improving the overall security of the IoT network.

2) Optimization of RGRU using ACOA

By optimizing important hyperparameters, such as learning rate and GRU neuron count, ACOA improves the performance of the RGRU neural network. This optimization makes the RGRU more effective and adaptable, improving its capacity to identify and analyze intricate patterns in sequential data. ACOA improves the RGRU configuration by imitating the adaptive hunting and defensive systems of coatis, inspired by their strategic behaviors. This ensures that ACOA has better learning and prediction capabilities. Table I shows the hyperparameters and values for GRU optimization.

TABLE I. HYPERPARAMETERS AND VALUES FOR GRU OPTIMIZATION

Symbols	Hyperparameters	Values
L	Learning Rate	[.01,.05,.1,.2,.5,.8,1.0]
G_N	GRU neuron count	[64,128,256,512,1024]

a) Initialization

To tune ERGRU, key hyperparameters were adjusted such as learning rate (L) and GRU neuron count (G_N). To start the optimization process, a candidate solution is generated, which is represented by the number of coats in the search space. The values of the choice factors are determined by each coati's position in this area. The coatis' locations in the search space are initialized using (8) at the start of the ACOA implementation.

$$C = \{c_1, c_2, c_3, \dots \dots, c_n\} \quad (8)$$

where C is the population and c_n represents the n^{th} solution.

$$c_1 = [L, G_N] \quad (9)$$

b) Fitness Calculation

The fitness of each initialized solution is calculated. Fitness is determined by calculating accuracy, with higher accuracy indicating better fitness. Equation (10) shows the fitness function.

$$Fitness = Max(Accuracy) \quad (10)$$

Accuracy is calculated by:

$$Accuracy = \frac{TN+TP}{TP+FP+TN+FN} \quad (11)$$

c) Update the Solution

In their native environments, coatis perform two unique actions that inspire the ACOA: hunting lizards and protecting themselves from predators. By considering these behaviors, potential solutions are modified. Half of the coatis climb trees after a lizard falls, while the other half stay on the ground throughout the phase of exploration (attacking behavior). The coatis's position on the tree can be determined using:

$$S_i^{p1}: s_{i,j}^{p1} = s_{i,j} + r \cdot \left(iguana_j - I \cdot s_{i,j}\right)$$

$$i = 1,2,\ldots\ldots,\left[\frac{N}{2}\right], \quad j = 1,2,\ldots.m \tag{12}$$

The following formula can be used to locate a spiny-tailed lizard that has fallen to the ground:

$$iguana_j^k = l\beta_j + r \cdot \left(u\beta_j - \beta_j\right) \tag{13}$$

$$S_i^{p1} = \begin{cases} s_{i,j} + r \cdot \left(iguana_j^k - I.s_{i,j}\right), & f_{iguana}^k < f_i \\ s_{i,j} + r \cdot \left(s_{i,j} - iguana_j^k\right), & \text{else} \end{cases} \tag{14}$$

where:

$$i = \left[\frac{N}{2}\right] + 1, \left[\frac{N}{2}\right] + 2,\ldots\ldots,N, \quad j = 1,2,\ldots,m.$$

In this formula, S_i denotes as the i^{th} coati in the search space, and $s_{i,j}$ denotes the value of the j^{th} decision variable. The total number of coatis is denoted as N, the number of decision variables is denoted as m, a random real number between 0 and 1 is denoted as r, and $l\beta_j$ and $u\beta_j$ denote the lower and upper limits of the j^{th} decision, respectively.

$$S_i = \begin{cases} S_i^{p1} & f_i^{p1} < f_i \\ S_i & else \end{cases} \tag{15}$$

The new position for the i^{th} coati is estimated using S_i^{p1}, where $s_{i,j}$ represents the j^{th} dimension of the i^{th} coati's position. The variable $s_{i,j}^{p1}$ indicates the j^{th} dimension, f_{iguana}^k represents the value of the objective function, and r is a random real number within the range [0,1]. The position of the spiny-tailed lizard in the search space, which indicates the location of the best member, is indicated by $iguana_j$, where j denotes the j^{th} dimension. Additionally, I is an integer randomly chosen from the set {1,2}, $\lfloor \cdot \rfloor$ is the floor function, $iguana^k$ depicts the lizard's position on the ground, and $iguana_j^k$ indicates the j^{th} dimension of the lizard's location.

d) Levy Flight Mechanism

The algorithm's flexibility and convergence speed are improved when adaptive weighting and the Levy flight mechanism are incorporated during the exploration stage. This results in a notable enhancement in the algorithm's search capabilities. The following formula is used for this improvement:

$$\omega(t) = 0.2\cos\left(\frac{\pi}{2}\left(1 - \frac{t}{T}\right)\right) \tag{16}$$

$$S_i^{t+1} = S_i^t + r_1 + L \oplus Lery(\lambda) \tag{17}$$

where S_i^t denotes the location of the i^{th} coati during t iteration, T is the maximum number of iterations, and L denotes the step size control parameter. When specifying a specific path or trajectory, $L = 0.01\left(S_i^t(t) - S_p\right)$, and $Lery(\lambda)$ indicates the Levy distribution.

e) Exploitation Phase (Escaping Behavior)

In ACOA, optimization is guided by a behavior inspired by the survival instinct of coatis, prompting them to flee from predators.

$$S_i^{p2}: s_{i,j}^{p2} = s_{i,j} + (1 - 2r) \cdot \left(l\beta_j^{local} + r \cdot \left(u\beta_j^{local} - l\beta_j^{local}\right)\right) \tag{18}$$

where $l\beta_j^{local} = \frac{l\beta_j}{t}$, $u\varphi_j^{local} = \frac{u\beta_j}{t}$, S_i^{p2} is the modified spot of the i^{th} lengthy-nosed coatis, $s_{i,j}^{p2}$ is the j^{th} dimension of the coatis, f_i^{p2} is the coatis's objective function value, r is a randomly chosen real number between 0 and 1, and $l\beta_j$ and $u\beta_j$ stand for the lower and upper boundaries of the j^{th} decision variable.

f) Termination

The ACOA iteration is considered completed when there is a shift in location for each coati in the search space between phases one and two. Applying (12)-(18), the population is updated iteratively until the last iteration is reached. As a result of optimization, the ERGRU's performance improves and provides more accurate results. The learning rate and number of GRU neurons are two essential hyperparameters that ACOA carefully optimizes.

C. Cryptographic Hash Generation using Miyaguchi–Preneel Technique

After classifying the data, the proposed method ensures secure transfer by utilizing blockchain technology based on the Merkle-Damgård cryptographic hash. The usage of blockchain technology protects against unauthorized access to data transfers.

In the construction of the blockchain, several blocks are connected to form a chain. Each block in the chain consists of a timestamp (t_s), data (h_i), and a cryptographic hash of the previous block ($Prev_{hash}$). Every user on the blockchain has a unique transaction history. Sensitive data collected by sensor nodes is part of every transaction. The Merkle-Damgård hash cryptographic function is utilized to ensure data preservation. Every transaction generates a blockchain using the Merkle–Damgård hash cryptographic technique. The hash function divides the data into message blocks and a hash value is created for every message block. Sensitive data is provided to the server with its final hash value to improve data security. The prior block's hash (prev-hash) is used for block confirmation. The time steps (tsp) show the time at which the block was formed.

The Merkle-Damgård architecture uses a one-way compression algorithm to build a fixed hash for each data input. Before creating the hash, the Merkle-Damgård construction divides the input data into fixed-size message blocks.

$$h_s \rightarrow M_1, M_1, M_1, M_r$$

After the data is split, the message block is passed to the compression function O_{fr}, which requires a final hash, i.e., an $r - bit$ message block M_r and an $r - bit$ chaining value Φ_a. Using compressive Merkle-Damgård construction, a fixed-length message is processed iteratively by the compression function $O_{f1}, O_{f2}, .O_{fr}$, resulting in the hash number $\Phi_a \in \{0,1\}$. The compression function is provided using a

straightforward block cipher method. For the lone output, the hash is the same size as the input hash. The two fixed-size inputs that are formed are the message block and the previous hash.

An initial hash of the algorithm is denoted as Φ_0:

$$\Phi_0 = O_F[\Phi_{i-1}, M_i], \quad i = 1,2,3,\ldots r \qquad (19)$$

The data's final hash value is Φ_a , the function of compression is O_F, the previous block hash is Φ_{i-1} and the block of messages is M_i.

III. RESULTS AND DISCUSSION

The proposed system was implemented on a Windows 10 PC with an i5 processor running at 1.6 GHz and 16 GB of RAM using Python. The DDoS dataset in [19] was used to evaluate the effectiveness of the proposed method.

A. 3.1 Experimental Results

The proposed approach was compared with the LSTM, GRU, and Bidirectional Gated Recurrent Unit (BiGRU) models using various metrics. In addition, the proposed system's performance evaluation examined processing speed, data integrity, and confidentiality rates. The dataset was divided into training and testing validation ratios of 70:30 and 80:20. Table II presents a comparison of various methods for the 70:30 data split, evaluated across different metrics. Table III provides a similar comparison for the 80:20 data split.

TABLE II. COMPARISON TABLE WITH 70:30 SPLIT

Methods	GRU	LSTM	Bi-GRU	Proposed
Accuracy	0.96712	0.95665	0.96081	0.97925
Precision	0.96701	0.95645	0.96065	0.97915
F-score	0.96805	0.95830	0.96218	0.97002
Specificity	0.96812	0.95843	0.96228	0.97008
Sensitivity	0.96793	0.95810	0.96201	0.97992
MCC	0.96797	0.95816	0.96206	0.97995
NPV	0.96805	0.95830	0.96218	0.97002
FPR	0.05765	0.06416	0.05965	0.04614
FNR	0.03787	0.04622	0.04106	0.03565

The findings in Table II show that the proposed ERGRU model achieved the best performance, with an accuracy of 97.93%. It also demonstrates the highest precision of 97.92%, F-score of 97.00%, specificity of 97.01%, sensitivity of 97.99%, MCC of 97.99%, NPV of 97.00%, FPR of 0.04614%, and FNR of 0.03565%, indicating its ability to effectively detect both normal and attack instances. BiGRU and GRU follow with an accuracy of 96.08% and 96.71%, commendable precision of 96.06% and 96.70%, F-score of 96.22% and 96.80%, sensitivity of 96.20% and 96.79%, specificity of 96.23% and 96.81%, MCC of 96.20 and 96.80%, NPV of 96.22% and 96.80%, FPR of 0.05965% and 0.05765%, and FNR of 0.04106% and 0.03787%, respectively. The LSTM method demonstrated slightly lower accuracy levels of 95.66%, precision of 95.64%, F-score of 95.83%, specificity of 95.84%, sensitivity of 95.81%, MCC of 95.81%, NPV of 95.83%, FPR of 0.06416%, and FNR of 0.04622%, respectively. For the 70:30 data split, all models showed good predictive performance and high accuracy. However, the proposed model performed significantly better in most measures, suggesting its efficacy for the task at hand.

TABLE III. COMPARISON TABLE WITH 80/20 SPLIT

Methods	LSTM	Bi-GRU	GRU	Proposed
Accuracy	0.97046	0.97458	0.97872	0.98712
Precision	0.97021	0.97436	0.97854	0.98701
F-score	0.97255	0.97638	0.98024	0.98805
Specificity	0.97271	0.97652	0.98035	0.98812
Sensitivity	0.97229	0.97615	0.98005	0.98793
MCC	0.97236	0.97622	0.98010	0.98797
NPV	0.97255	0.97638	0.98024	0.98805
FPR	0.05660	0.04762	0.04110	0.03614
FNR	0.03614	0.03226	0.02913	0.02655

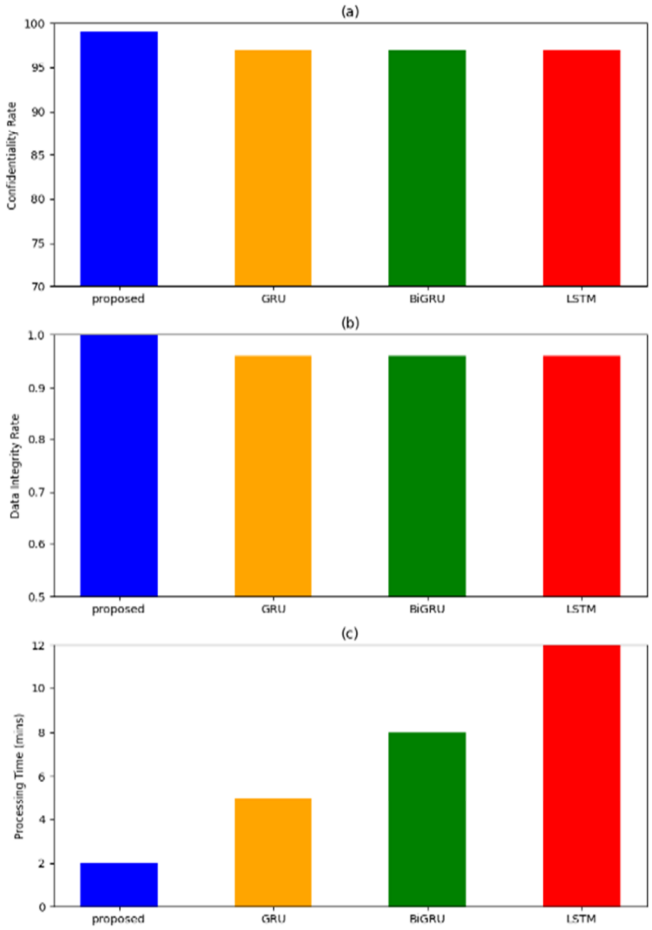

Fig. 3. (a) Confidentiality rate, (b) integrity rate, (c) processing time.

The findings in Table III show that the proposed proposed ERGRU model achieved the most superior performance, with an accuracy of 98.71%. It also demonstrated the highest precision of 98.70%, F-score of 98.80%, specificity of 98.81%, sensitivity of 98.79%, MCC of 98.97%, NPV of 98.80%, FPR of 0.03614%, and FNR of 0.02655%, indicating its ability to effectively detect both normal and attack instances. Comparatively, GRU and BiGRU followed with an accuracy of 97.87% and 97.45% and exhibited commendable precision of 97.85% and 97.44%, F-score of 96.22% and 96.80%, the

sensitivity of 98.00% and 97.61%, specificity of 98.03% and 97.65%, MCC of 98.01% and 97.62%, NPV of 98.02% and 97.63%, FPR of 0.04110% and 0.047625%, and FNR of 0.02913% and 0.03226%. The LSTM method demonstrated a slightly lower accuracy of 97.04%, precision of 97.02%, F-score of 97.25%, specificity of 97.27%, sensitivity of 97.29 %, MCC of 97.24%, NPV of 97.25%, FPR of 0.05660%, and FNR of 0.03614%, respectively. For the 80:20 ratio data split, all models showed good predictive performance and high accuracy. However, the proposed model performed significantly better across most measures, suggesting its efficacy for the task at hand. Compared to existing models, the proposed approach demonstrated superior results, indicating its efficacy against DDoS threats.

Figure 3 shows the data integrity rate, the processing time, and the confidentiality rates. High confidentiality, data integrity rate, and processing time demonstrate how well the proposed system protects against illegal access and data tampering. The proposed DDOS mitigation strategy achieved significant performance metrics, such as a 100% data integrity rate and a 98.8% confidentiality rate, coupled with a 2-minute processing time. These numbers highlight how well the proposed model handles DDoS attack detection and mitigation while maintaining data confidentiality and integrity.

IV. CONCLUSION

Strong security measures are increasingly necessary to protect sensitive data from potential dangers as the number of IoT devices increases, particularly in the context of DDoS attacks. This study proposed a unique strategy that used blockchain technology by combining smart contracts with a sophisticated attack detection system. An improved RGRU architecture was proposed for IoT network DDoS attack detection and mitigation. To further increase detection accuracy, the hyperparameters of the RGRU model were optimized using ACOA. Furthermore, the proposed framework utilized a one-way compression function to generate secure hashes for input data utilizing the Merkle-Damgård cryptography technique to ensure data integrity and confidentiality. The proposed method outperformed existing methods and achieved the highest accuracy of 98.71%. Future studies might focus on expanding the framework to handle other types of cyber threats, such as Advanced Persistent Threats (APTs) and insider attacks.

REFERENCES

[1] J. Tournier, F. Lesueur, F. L. Mouël, L. Guyon, and H. Ben-Hassine, "A survey of IoT protocols and their security issues through the lens of a generic IoT stack," *Internet of Things*, vol. 16, Dec. 2021, Art. no. 100264, https://doi.org/10.1016/j.iot.2020.100264.

[2] "Number of connected IoT devices growing 13% to 18.8 billion." https://iot-analytics.com/number-connected-iot-devices/.

[3] I. Ali *et al.*, "Systematic Literature Review on IoT-Based Botnet Attack," *IEEE Access*, vol. 8, pp. 212220–212232, 2020, https://doi.org/10.1109/ACCESS.2020.3039985.

[4] S. Ghazanfar, F. Hussain, A. U. Rehman, U. U. Fayyaz, F. Shahzad, and G. A. Shah, "IoT-Flock: An Open-source Framework for IoT Traffic Generation," in *2020 International Conference on Emerging Trends in Smart Technologies (ICETST)*, Karachi, Pakistan, Mar. 2020, pp. 1–6, https://doi.org/10.1109/ICETST49965.2020.9080732.

[5] B. Ghimire and D. B. Rawat, "Recent Advances on Federated Learning for Cybersecurity and Cybersecurity for Federated Learning for Internet of Things," *IEEE Internet of Things Journal*, vol. 9, no. 11, pp. 8229–8249, Jun. 2022, https://doi.org/10.1109/JIOT.2022.3150363.

[6] K. S. Kumar, S. Alqarni, S. Islam, and M. A. Shah, "Royal Poinciana Biodiesel Blends with 1-Butanol as a Potential Alternative Fuel for Unmodified Research Engines," *ACS Omega*, vol. 9, no. 12, pp. 13960–13974, Mar. 2024, https://doi.org/10.1021/acsomega.3c09014.

[7] Kamaldeep, M. Malik, and M. Dutta, "Feature Engineering and Machine Learning Framework for DDoS Attack Detection in the Standardized Internet of Things," *IEEE Internet of Things Journal*, vol. 10, no. 10, pp. 8658–8669, Feb. 2023, https://doi.org/10.1109/JIOT.2023.3245153.

[8] F. Hussain, S. G. Abbas, M. Husnain, U. U. Fayyaz, F. Shahzad, and G. A. Shah, "IoT DoS and DDoS Attack Detection using ResNet," in *2020 IEEE 23rd International Multitopic Conference (INMIC)*, Bahawalpur, Pakistan, Nov. 2020, pp. 1–6, https://doi.org/10.1109/INMIC50486.2020.9318216.

[9] Y. N. Soe, Y. Feng, P. I. Santosa, R. Hartanto, and K. Sakurai, "Machine Learning-Based IoT-Botnet Attack Detection with Sequential Architecture," *Sensors*, vol. 20, no. 16, Jan. 2020, Art. no. 4372, https://doi.org/10.3390/s20164372.

[10] N. Vlajic and D. Zhou, "IoT as a Land of Opportunity for DDoS Hackers," *Computer*, vol. 51, no. 7, pp. 26–34, Jul. 2018, https://doi.org/10.1109/MC.2018.3011046.

[11] M. Aslam, D. Ye, M. Hanif, and M. Asad, "Machine Learning Based SDN-enabled Distributed Denial-of-Services Attacks Detection and Mitigation System for Internet of Things," in *Machine Learning for Cyber Security*, Cham, 2020, pp. 180–194, https://doi.org/10.1007/978-3-030-62223-7_16.

[12] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, Sep. 2013, https://doi.org/10.1016/j.future.2013.01.010.

[13] R. Taylor, D. Baron, and D. Schmidt, "The world in 2025 - predictions for the next ten years," in *2015 10th International Microsystems, Packaging, Assembly and Circuits Technology Conference (IMPACT)*, Taipei, Taiwan, Oct. 2015, pp. 192–195, https://doi.org/10.1109/IMPACT.2015.7365193.

[14] S. K. Dash *et al.*, "Enhancing DDoS attack detection in IoT using PCA," *Egyptian Informatics Journal*, vol. 25, Mar. 2024, Art. no. 100450, https://doi.org/10.1016/j.eij.2024.100450.

[15] H. Al-Hamadi, I.-R. Chen, D.-C. Wang, and M. Almashan, "Attack and Defense Strategies for Intrusion Detection in Autonomous Distributed IoT Systems," *IEEE Access*, vol. 8, pp. 168994–169009, 2020, https://doi.org/10.1109/ACCESS.2020.3023616.

[16] F. Hussain *et al.*, "A Two-Fold Machine Learning Approach to Prevent and Detect IoT Botnet Attacks," *IEEE Access*, vol. 9, pp. 163412–163430, 2021, https://doi.org/10.1109/ACCESS.2021.3131014.

[17] J. Bhayo, S. Hameed, and S. A. Shah, "An Efficient Counter-Based DDoS Attack Detection Framework Leveraging Software Defined IoT (SD-IoT)," *IEEE Access*, vol. 8, pp. 221612–221631, 2020, https://doi.org/10.1109/ACCESS.2020.3043082.

[18] Y. Alhasawi and S. Alghamdi, "Federated Learning for Decentralized DDoS Attack Detection in IoT Networks," *IEEE Access*, vol. 12, pp. 42357–42368, 2024, https://doi.org/10.1109/ACCESS.2024.3378727.

[19] "DDoS Dataset." Kaggle, [Online]. Available: https://www.kaggle.com/datasets/devendra416/ddos-datasets.