Real-Time Attack Detection and Prevention in Bluetooth Communication: Evaluating Encryption Strategies

Soma Siva Pravallika
Department of Computer Science and Engineering
Amrita Vishwa Vidyapeetham
Ettimadai, Coimbatore, India
cb.en.u4cse22440@cb.students.amrita.edu

Sribhashyam Indira Satya Sri
Department of Computer Science and Engineering
Amrita Vishwa Vidyapeetham
Ettimadai, Coimbatore, India
cb.en.u4cse22441@cb.students.amrita.edu

R Rinitha
Department of Computer Science and Engineering
Amrita Vishwa Vidyapeetham
Ettimadai, Coimbatore, India
cb.en.u4cse22556@cb.students.amrita.edu

Aparna Srinath
Department of Computer Science and Engineering
Amrita Vishwa Vidyapeetham
Ettimadai, Coimbatore, India
s_aparna@cb.amrita.edu

***Abstract*—In recent years with the rapid increase of cyber threats, Bluetooth has become one of a primary target for attacks like Man-in-the-Middle (MitM), DDoS, and device spoofing which exploit the vulnerability in pairing protocol, encryption scheme, and authentication mechanisms. To mitigate these security gaps, it is imperative to improve the effectiveness of the Bluetooth attack detection and encryption mechanism. In this article, we present a novel approach with a dual-focused solution, (1) a lightweight Graph Convolutional Network (GCN) model to detect attacks in Bluetooth communication and (2) a comparison of five lightweight encryption algorithms: ChaCha20, AES- CCM, Ascon, PRESENT, and SPARX to determine the most efficient model for secure Bluetooth communication in a resource constraint environment. We evaluated the performance of the proposed model and achieved improved accuracy on our custom generated dataset and a pre-existing dataset as well. To further reinforce security, a preventive method was implemented to monitor the request frequency of each device and block addresses that exhibit unusual behavior.**

***Index Terms*—Lightweight Encryption, ESP32 Communication, Bluetooth Packet Analysis, Graph Neural Network(GNN), Bluetooth Security.**

I. Introduction

Edge computing is a way of doing work that provides a level of computing that is more local and less reliant on cloud computing. Edge computing is helping smart devices and the Internet of Things (IoT) to perform real-time data analysis with reduced latency and bandwidth use. It provides seamless functionality and is part of IoT. However, connection processes such as Bluetooth pairing can be hacked. This paper will discuss these detection of attacks on bluetooth pairing, as well as ways to safeguard and prevent attacks.

Edge computing does have some security issues. The attack surface grows in edge computing, making the devices more vulnerable to attack. The lack of strong passwords allows intruders to gain unauthorized entry into IoT devices. Ensuring security policies across all devices is hard and complex as so many edge devices with different security requirements pose scalability problems. To ensure consistent and accurate data processing across multiple edge nodes, we need mechanisms that can guarantee the authenticity of data and prevent tampering.

One such example of a security vulnerability in edge computing involves Bluetooth communication which is vulnerable to passive eavesdropping, where an attacker intercepts unencrypted data transmitted between a smartwatch and a paired device. This attack exploits weaknesses in older Bluetooth versions that lack proper encryption mechanisms. Sensitive information such as health data, notifications, and authentication tokens can be compromised. Secure pairing methods like SSP (Secure Simple Pairing) help mitigate this risk. Using Bluetooth Low Energy (BLE) encryption further enhances security against interception.

Common bluetooth communication attacks are as follows Man-in-the-Middle (MITM) attack that occurs when an ad- versary secretly intercepts and alters the data exchanged be- tween ESP32 and a smartphone. Weak or poorly implemented authentication protocols make this attack feasible. Attackers can manipulate transmitted commands, extract sensitive data, or inject malicious instructions. Secure pairing mechanisms, such as numeric comparison or passkey entry, help prevent MITM attacks. Ensuring end-to-end encryption in Bluetooth communication strengthens defense against unauthorized data manipulation. A Bluetooth-based Denial of Service (DDoS) attack disrupts the functionality of the server by overwhelming it with excessive connection requests. Attackers may exploit Bluetooth protocols to repeatedly send pairing requests, draining the life of battery and causing system crashes. This can

temporarily render the smart device unusable, affecting critical functionalities. Boosting limiting Bluetooth requests and implementing timeout mechanisms help mitigate the DDoS attack. Advanced Bluetooth versions with enhanced security policies offer additional resilience against such threats. Bluetooth MAC spoofing involves changing a device's MAC address to bypass security measures and appear as a trusted entity. Attackers exploit this technique to gain unauthorized access to paired devices. This method is particularly effective in environments where MAC-based authentication is the primary security mechanism. Implementing multifactor authentication and dynamic MAC address randomization reduces susceptibility to spoofing. The enforcement of additional authentication layers strengthens Bluetooth security against impersonation. The device impersonation attack is an adversary mimics a smart device to trick users into pairing with a malicious device. Attackers can use spoofed Bluetooth device names or MAC addresses to appear trustworthy. Once paired, they may extract sensitive data or deploy malicious software. Preventing automatic pairing and verifying device identities through authentication codes reduces impersonation risks. Using Bluetooth security features such as Identity Resolving Keys (IRK) helps detect and prevent unauthorized pairing.

Encryption ensures data confidentiality by transforming plain text into an unreadable format which prevents unauthorized access and modification during data transmission. Implementing lightweight encryption algorithms is essential for resource-constrained devices like ESP32. ChaCha20 is a lightweight, designed for high-speed encryption with strong security guarantees. Using a series of mathematical operations, it generates a key of 256-bit key and a 128-bit nonce. It offers resistance against timing attacks and efficient performance on resources-constrained devices such as ESP32. Advanced Encryption Standard (AES) in conjunction with the Cipher Block Chaining Message Authentication Code (CCM) mode is a widely used encryption scheme combining confidentiality and authentication. AES-CCM is used over AES-GCM (Galois / Counter Mode) for its lower computational overhead, optimized for resource-constrained environment, and usage in updated BLE version. Ascon is a lightweight authenticated encryption algorithm for efficiency and bluetooth security applications in embedded systems that uses a special design called a sponge function that is mainly used with limited computational resources, low latency, and energy consumption. PRESENT is an ultralightweight block cipher designed for minimal hardware and energy efficiency. It operates on a 64-bit block size with key sizes of 80 or 128 bits, which uses a substitution-permutation (SPN) structure to ensure security. SPARX is a lightweight block cipher designed for security and performance in constrained environments. It uses an extended substitution-permutation network (ESPN) and supports key sizes of 128 and 256 bits, which offers resistance against linear and differential cryptanalysis with low memory usage.

In this paper, we tackle the security issues of Bluetooth communication as identified by presenting a two-fold approach. To begin with, we compare and analyze five lightweight encryption schemes (ChaCha20, AES-CCM, Ascon, PRESENT, and SPARX) on the ESP32 platform and identify the most efficient solution to secure Bluetooth communication. Second, we propose a Graph Convolutional Network (GCN) model to identify and classify attacks on Bluetooth communication, namely Man-in-the-Middle (MitM) and Denial-of-Service (DDoS) at- tacks. We prove our method to be effective through enhanced accuracy in attack detection and the application of a preventive approach to counter MITM and DDoS attacks. In this we sent a message from phone to ESP32 and captured that corresponding packet in wireshark, sent to GCN model for testing. If we get MITM attack we follow encryption strategy to mitigate.

II. RELATED WORKS

In this section, we discuss previous studies and research pertaining to our work. These include the increasing security issues in IoT, intrusion detection using machine learning and deep learning, and the application of Graph Neural Networks (GNNs) to improve IoT security solutions. Further, we talk about the vulnerabilities of Bluetooth Low Energy (BLE) devices and different cryptographic protocols and mention how our work extends the scope of previous work to enhance real-time attack detection and mitigation within IoT networks.

A. Detection

The Internet of Things (IoT) evolved at a breakneck speed penetrating many sectors such as healthcare, smart cities, industrial automation, and home automation. However, the increase in IoT devices has also posed serious security issues, most importantly intrusion detection. IoT networks are greatly susceptible to cyber attacks because they are distributed networks, resource-limited, and without adequate security mechanisms. Different research works attempted machine learning (ML), deep learning (DL), and graph-based approaches to design effective Intrusion Detection Systems (IDS). Although they have made excellent contributions towards anomaly detection, it is difficult to suppress false alarms, stay responsive to new attack patterns, and stay computationally efficient in low- resource settings. New developments in Graph Neural Networks (GNNs) and learning algorithms have opened new ways to further improve the accuracy, scalability,and real- time processing of IoT security solutions. Our implementa- tion which is GCN based security approach enhances real- time detection while maintaining computational efficiency for resource-constrained IoT devices.

The security landscape of IoT networks has been a topic of interest in research due to the increasing threat of sophisticated attacks. Various research studies have addressed various problems of IoT security, including intrusion detection, cryptographic protocols, weaknesses of Bluetooth Low Energy (BLE) and graph convolutional networks (GCNs) for anomaly detection. The current section offers a detailed review of the literature and maps it to our implementation, which integrates GCN-based security aspects into IoT frameworks. Our proposed system leverages GCNs to process network

traffic data and effectively distinguish between normal and malicious activities, providing a more robust intrusion detection mechanism.

IoT networks are also extremely susceptible to many forms of cyber attacks such as Denial-of-Service (DoS) attacks, Man-in-the-Middle (MITM) attacks, and packet sniffing. Various researchers have studied the security issues of IoT devices. Various security issues in IoT devices have been discussed by Mary et al., and it is necessary to use robust authentication and encryption measures [1]. In addition, B.Zhang et al. sug- gest AI-based intrusion detection systems to detect malicious activity in IoT networks through machine learning algorithms [2]. The study indicates the need for sophisticated detection mechanisms, which our research is based on a GCN-based method for real-time anomaly detection.

Hlapisi [3] provides a comprehensive overview of security threats targeting Bluetooth Low Energy (BLE) devices, highlighting vulnerabilities such as key reuse, weak pairing methods, and exposure to sniffing, replay, and man-in-the-middle (MITM) attacks. The study emphasizes the critical need for strong encryption, secure authentication, and timely firmware updates to mitigate these risks Machine learning approaches have been widely utilized for the purpose of intrusion detection in IoT networks. Traditional ML methods like Support Vector Machines (SVM), Decision Trees (DT) [4], and Random Forest (RF) have been thoroughly used to classify traffic in a network as malicious or benign. These methods greatly rely on hand-designed feature extraction methods, which are highly domain-specific and need immense domain knowledge, and are often designed for specific categories of attacks. However, their inability to move makes them less effective against zero- day attacks and creating new versions of known threats. Our approach addresses this limitation by employing GCNs, which can dynamically learn network structures and relationships rather than relying on predefined feature extraction.

Deep learning methods are now dominant in intrusion detection because they have the ability to learn network spatial and temporal traffic patterns automatically.The use of machine learning algorithms for detecting DDoS attacks, such as the XGBoost algorithm as one of the the primary methods is to be noted [7]. Convolutional Neural Networks (CNNs) [8] are widely utilized to detect anomalies with the capability to learn spatial properties from network traffic. Additionally, Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks provided improved performance to sequence-based anomaly detection with temporal patterns of network traffic learning capabilities. Autoencoders (AE) [11] have also been used to identify abnormality from typical network behavior by reconstructing the input data and marking as anomalies high reconstruction errors. A lightweight, rule-based firewall designed specifically for Bluetooth Low Energy (BLE) communication would enhance security against threats like spoofing and unauthorized access. Ahmed Aboukora et al. implemented using Python and a BLE sniffer, the system monitors and filters BLE packets in real-time without modifying the BLE stack [9]. This approach offers an efficient and non-intrusive security layer suitable for resource-constrained IoT environments Our method, however, goes further by using GCNs, which are especially well-suited for dealing with network-structured data, enabling us to capture interactions and dependencies among IoT devices more effectively.

B. Prevention

Bluetooth Low Energy (BLE) devices, commonly utilized across IoT networks, have been the primary focus of attacks since they do not have strict security controls. Bluetooth LE device attacks and threats require a significant contemporary concern.Bluetooth Low Energy (BLE) has emerged as a widely adopted communication protocol for IoT and wearable devices due to its low power consumption and efficient data transfer capabilities [14], and network-based threat detection is has also escalated. Post-Quantum Cryptography (PQC) has gained significant attention due to the potential threats posed by quantum computing to classical cryptographic systems. Various works have explored the challenges and opportunities of implementing PQC in resource-constrained Internet of Things (IoT) devices[16].Sometimes, the proposed pairing process security weaknesses seem a major concern. Therefore, the detailed study of MITM attacks on Bluetooth smart devices and counter measures how to hamper them is crucial. Our work extends these studies by utilizing GCN-based network analysis to identify abnormal BLE communication patterns and possible attack vectors.

Several real-world attacks have successfully demonstrated the exploitability of BLE vulnerabilities. R. Kaur et al investigates network traffic using packet sniffing tool such as Wireshark [18]. The ”BlueTack” attack where attackers intercept and tamper with Bluetooth BR/EDR communica- tion , Wireshark can be used to investigate them. A countermeasure deployed in order to combat Man-in-the-Middle (MITM) attacks for Bluetooth Low Energy (BLE) includes packet filtering and blocking unauthorized devices. Through this mechanism, verified devices only get to form connec- tions while observing incoming BLE packets for suspicious patterns or spoofed credentials. Upon detection of attempts to access in unauthorized manners, the system is capable of disrupting connections and hindering communication fur- ther, thus securing data integrity and confidentiality within IoT ecosystems.Moreover,the BLUR-tooth attacks showed that Bluetooth communication can be attacked by cross-transport key derivation . These findings are applied in our study with GCNs to analyze network traffic patterns and detect possible MITM attacks within IoT networks. By combining these findings, our system improves detection and prevention of BLE-specific attacks through graph-based anomaly detection and we have used wireshark to retrieve the bluetooth packets as demonstrated in *Figure 5*.

Graph-based learning models as GCNs have received immense interest because of their ability to learn high-order relationships in network traffic data. Recent research has shown the promise of GCNs in anomaly detection and intrusion prevention. This paper builds on existing GCN-based security

TABLE I
COMPARATIVE ANALYSIS OF BLUETOOTH ATTACK DETECTION AND ENCRYPTION APPROACHES

Reference	Encryption Algorithm	Attack Type			Evaluation Metrics	Dataset Used	Detection Method
		MitM	DDoS	Spoofing			
S. S. Mary [1]	AES-CCM	●	❍	●	Security, Stability (LE 73%), MITM Prevention (99.7%)	IEEE Bluetooth Security Data	AI, Wireshark, Bettercap, BtleJuice, Crackle
B. Zhang [2]	AES-CCM	●	❍	●	Accuracy (>90%), Low False Positive Rate	Custom Bluetooth Signal Dataset	Graph Neural Network (GNN)
Nthatisi Hlapisi [3]	AES-CCM	●	❍	●	Throughput, Energy	BLE Attack Dataset	Deep Learning
Ray Felch [5]	AES-CCM	●	❍	❍	Practical Feasibility, Portability, Cost-effective (<$100)	Not Applicable (Live PoC)	Raspberry Pi + BTD-400-based MITM Framework
T Melamed [6]	ECDH	●	❍	❍	Not explicit quantified	A practical demonstration, but no dataset	Tool-based (GATT) Practical Attack Simulation
Ahmed Aboukora, [9]	Standard BLE encryption	●	❍	❍	Packet Filtering Capability, Service/Characteristic Hiding, Pairing Enforcement	live traffic and scenario based	real-time packet inspection and filtering (GATT level)
M. TajDini [10]	AES-128	●	❍	❍	Success rate of the MitM attack,Packet interception and relay capability,Effectiveness of jamming and channel hopping,Practicality in a real-world setting using hardware like HackRF One	Custom BLE traffic generated between a BLE fitness tracker and a smartphone, and then intercepted using SDR-based tools.	Software: GNU Radio, Wireshark, Gatttool Hardware: HackRF One (Software Defined Radio), BLE devices (e.g., Xiaomi Mi Band 3) Techniques used: BLE advertisement sniffing, jamming, replay, and relaying
A.Hekmati, B.Krishnamachari [26]	Not specified (Focus on DDoS detection rather than encryption)	❍	●	❍	F1-Score (up to 91%), Resilience to 50% connection loss (less than or equal to 2% drop in F1)	Simulated IoT traffic under lossy conditions.	Graph Convolutional Network (GCN)-based Detection

solutions by integrating them into IoT security systems for improved attack detection and mitigation. Our implementation improves IoT network security by exploiting GCNs for anomaly detection, enabling more adaptive response to evolving attack patterns and protecting IoT communication channels.

References [2],[37],[38],[39],[40] refer to GCN applications in cybersecurity, their value for real-time IoT communication graph anomaly detection and modeling. Zhang introduces an authentication scheme in IoT device secure authentication using graph-based learning approaches [2]. Kipf and M. Welling, examine cyber attacks in wearable technology with a particular emphasis on the detection of passive attacks using GCN-based classifiers [37]. Z. Wu et al. examine GCN-based security frameworks for BLE devices with extreme accuracy in identifying spoofing attacks [38]. Finally, W. Hamilton et al. present a GCN-ML hybrid model for the security of IoT networks using a combination of graph-based learning and traditional ML classifiers for stronger threat detection [39]. The findings of these papers firmly affirm our GCN-based solution through highlighting the use of graph-based learning for IoT ecosystem cyber threat identification.

With the rise in cyber threats, Bluetooth has become a prime target for MitM, DDoS, and spoofing attacks due to vulnerabilities in pairing and encryption. To enhance security, various lightweight encryption algorithms have been explored. ChaCha20-Poly1305 is widely used for secure communication and its multi-user security has been analyzed [42]. AES- CCM remains a robust choice for authenticated encryption [10] in wireless networks [41]. Ascon, a NIST finalist, has been evaluated under fault attacks to ensure its resilience [44]. PRESENT, a hardware-efficient block cipher, is frequently referenced in IoT security research [45], while SPARX provides provable security bounds for ARX-based encryption [43].

This study compares these five encryption algorithms on ESP32 using metrics like decryption time, CPU cycles, memory usage, and throughput as illustrated in *Figure 6*. Additionally, a lightweight GCN model is developed to detect MitM, DDoS, and normal traffic as demonstrated in our architecture diagram *Figure 1*.

C. Encryption for IoT

Blockchain technology has been vowing to resolve the security issues in IoT via decentralized and tamper-evident authentication. Lee offers blockchain-based security mechanisms for IoT devices with an emphasis on secure key management [7].The use of AES-256 encryption to provide secure data transfer in IoT networks specifically for bluetooth connceted network devices is quite common as observed from Table I. Lightweight cryptography designed for IoT devices with limited capabilities, ensures security with efficiency . Blockchain architecture of IoT security combined with end-to-end encryption would thwart illegal access to information [7]. The survey paper [25] offers the first comprehensive taxonomy of BLE-related security and privacy issues. It categorizes

vulnerabilities (e.g., encryption flaws, weak authentication, privacy leaks).It also describes potential attack scenarios and classifies vulnerabilities by their severity and suggests appropriate mitigation strategies.It does include real-world case studies showing how these flaws can be exploited in existing BLE devices. The paper emphasizes the urgent need for addressing BLE vulnerabilities to secure the growing ecosystem of connected devices. Homomorphic encryption can be applied to secure communication in IoT and reduce the scope of data leakage . Our implementation aligns with these studies by incorporating encryption methodologies to complement the GCN-based anomaly detection system, ensuring that data remains secure even in the event of an intrusion attempt.

Several inclusive surveys provide an overview of BLE and IoT security threats, countermeasures, and trends whereas some research papers introduce an overall discussion on attack trends and defense mechanisms of IoT. The current study broadens these discussions by incorporating a GCN-based detection mechanism in addition to encryption protocols to achieve a multi-layered security for IoT networks. This paper extends such studies with the addition of a GCN-based security system designed particularly for real-time IoT threat detectio as illustrated in our architecture diagram *Figure 1*.

III. Methodology

This section explains how Bluetooth packets are captured, how the connection is secured with encryption algorithms, how data preprocessing is done, and how the graph model is used to detect attack. First, we've configured Wireshark and nRF Sniffer plugin to capture packets from ESP32 bluetooth. AES-CCM encryption was used to ensure secure communication between the client and server. Further details on the client and server.Once data is captured, the dataset can be preprocessed by feature scaling and splitting into training and testing. Then, a GCN model has been implemented for network traffic classification and attack detection. A detailed description of the model architecture, training procedures, and evaluation criteria was presented in this section.

The entire system's architecture has been shown in Figure 1, which illustrates the secured transmission of Bluetooth data after encryption, the interception of messages by intruders, packet capture and analysis. The GCN model is utilized to test the captured packets, which permits the classification of those packets as a DDoS attack, MITM attack or normal. Mitigation strategies are applied accordingly.

A. Capturing Bluetooth Packets of ESP32 Using Wireshark and nRF Sniffer

The Bluetooth communication vulnerabilities were analyzed with the help of the ESP32 device working as a smartwatch and the nRF Connect app on the mobile phone to send messages. We connected to the ESP32 by checking available Bluetooth devices and looking at its name which is advertised. After connecting, messages sent from the nRF Connect app were displayed on the OLED screen connected to the ESP32 via I2C.

By using Wireshark and the Nordic Packet Sniffer plugin, we show a passive attack to monitor LED messages for BLE while not altering any transmitted data. This way we could see how the packets were exchanged over BLE and if there were any weaknesses such as the data that were sent unencrypted and weak pairing methods, metadata exposure, etc.The setup had to have a lot of steps in order to have a successful capture environment. Initially, I installed Wireshark, ensuring that USB capture support was enabled, using its default settings. Afterward, we put the Nordic Packet Sniffer plugin ZIP file to download and unpack into a folder. Then we set up the Nordic development kit by connecting nRF52480 dongle to the PC.

The dongle was subsequently put into firmware update mode, and the relevant HEX file was selected and programmed utilizing the tool from Nordic. After checking that the firmware was updated successfully, the necessary plugin files were copied, the dependencies installed and Wireshark was configured with the Bluetooth sniffer interface.We ran a script in the Arduino IDE to get the ESP32 MAC address in order to check Bluetooth communication. Using the MAC address as a filter in Wireshark, we have captured Bluetooth packets corresponding to the ESP32.

B. Securing Bluetooth Communication with Encryption Algorithms

We aim to improve secure Bluetooth communication by implementing and evaluating various encryption algorithms, including AES-CCM, ChaCha20-Poly1305, SPARX, Ascon, and PRESENT. Our comparison focused on key performance metrics such as decryption time, CPU cycles, memory usage, packets received, and overall throughput. To demonstrate practical implementation, we developed a secure communication system between a client and a server over Bluetooth Low Energy (BLE) using AES-CCM. The system allows the client to select an appropriate AES key length (128-bit, 192-bit, or 256-bit) based on their security and performance requirements. Since AES-256 is more computationally intensive than AES-128 and AES-192, this flexibility helps balance security and efficiency considerations.

The client is allowed to choose the key length of the AES according to the confidentiality needs. The larger the key length, the more computer steps will be involved for encrypting data. When chosen, the AES key is converted from hex to byte form. AES-CCM is a type of encryption that uses a randomly generated 12-byte nonce. The encryption process gives us ciphertext along with a 16-byte authentication tag that verifies the message. Next, the client connects to the ESP32 server over Bluetooth Low Energy (BLE) using the MAC address. The server receives the encrypted message via a designated UUID of a GATT characteristic. On the server side, the ESP32 acts as a BLE server. It recevies the en- crypted packet from client and extracts authentication tag, ciphertext and nonce from the incoming data. The ESP32 utilizes the AES key associated with the message to perform decryption and an integrity check using the tag. Once verified successfully, there will be printed the decrypted message along

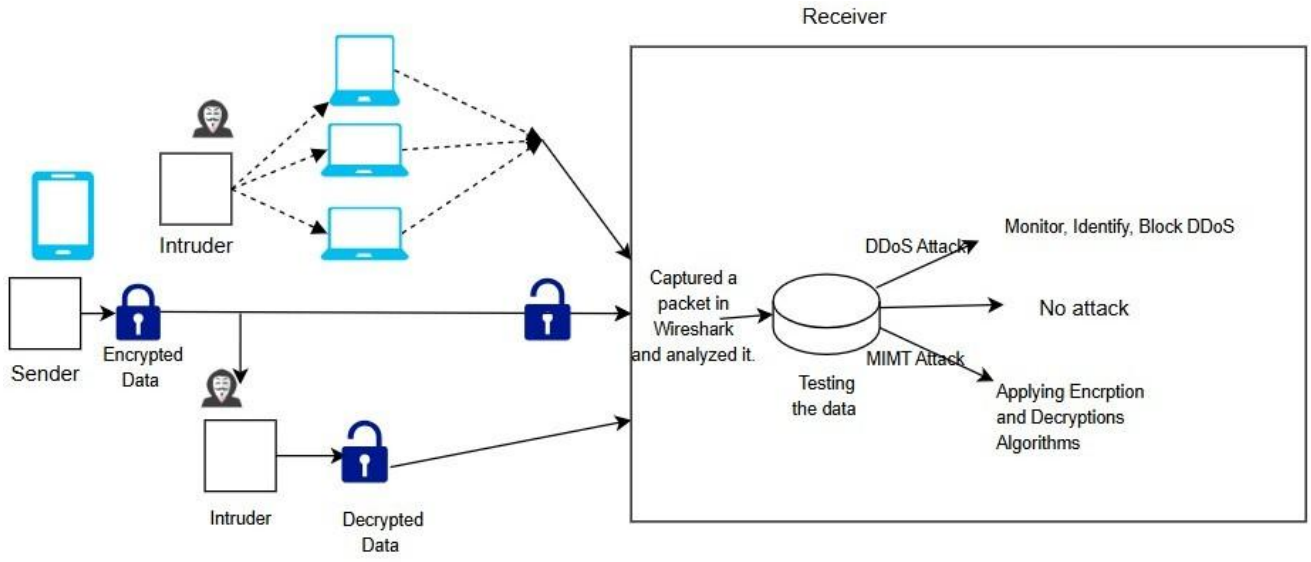

Fig. 1. Architecture Diagram

with the performance metrics like time taken to decrypt, number of CPU cycles used, memory used and throughput.

C. Dataset Preprocessing

[1]Dataset: https://ieee-dataport.org/documents/bluetack

The dataset we used in this study is Bluetooth-BR-EDR-BlueTack and DDoS Dataset.Bluetooth-BR-EDR-BlueTack, which includes network traffic data labeled as either Normal or Attack. This dataset specifically consists of popular attacks targeting Bluetooth BR/EDR (Bluetooth Classic) protocols, such as Bluesmack, DoS, DDoS, as well as attacks on the BLE protocol, including DDoS and MITM attacks. To en- sure consistency and improve model performance, the dataset undergoes feature scaling using Min-Max Scaling, which normalizes numerical attributes between 0 and 1. The dataset is split into 80% training and 20% testing to train and validate the model effectively.

The workflow of the GCN Model used in this work is illustrated in Figure 2. The workflow process involves dataset loading, data preprocessing, and graph construction. Once the GCN model architecture is defined, training occurs. Once the training is complete, the model is evaluated and the results are visualized to observe its effectiveness in Bluetooth-based attack detection.

D. Graph-Based Model Architecture

The model exploits a Graph Convolutional Network (GCN) for efficiently classifying network traffic into various attack classes. GCNs are well placed to do so as they have the ability to identify structural relationships between data points through propagating node features through graph edges. Two graph convolutional layers constitute our architecture which is

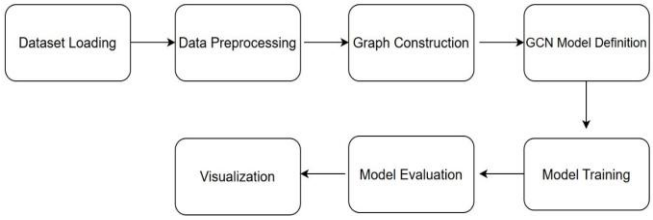

Fig. 2. GCN workflow diagram

used for obtaining meaningful representations out of the input graph. The first graph convolutional layer projects the input feature set onto a latent space allowing the model to learn complex patterns from network traffic data. A ReLU activation function is applied to introduce non-linearity, then dropout regularization with probability p=0.5 to prevent overfitting. The second graph convolutional layer maps the acquired feature representations to the target classification space,where each node receives a probability distribution over the possible attack types through a log-softmax activation.

The training process is executed for 50 epochs with the Adam optimizer having a learning rate of 0.01 and weight decay of 5e-4 to avoid overfitting. The data is divided into training (60The primary advantage of using a GCN-based approach is that it can diffuse and aggregate information using graph-structured data and is therefore extremely efficient in determining spatial and relational dependencies between different instances of network traffic. Through a two-layer architecture, expressiveness and computation are highly im- proved. Dropout regular-zation and weight decay improve generalization and prevent model overfitting to the training set. Model accuracy is benchmarked against a variety of

[1]

classification metrics such as accuracy, precision, recall, and F1-score, which give a general measure of its effectiveness in identifying malicious network behavior. With a graph-based learning process, our model is an extensible and robust solution to network intrusion detection. The ability to reveal latent relationships in the data increases predictive power, making it a very powerful tool for cybersecurity application. The Figure 3 yields a brief overview of the Graph Convolutional Network (GCN) layers used in our model. It explains the input layer and the hidden layer, explaining the process of feature aggregation which allows each node to collect feature information from its neighboring nodes with the help of the adjacency matrix.

In *Layer 0*, Input Layer, every node begins with an initial feature vector expressed in matrix form. The nodes are connected depending on their relations and thereby resulting in a graph structure. Moving to *Layer 1*, the first Hidden Layer employing GCNConv, the process of feature aggregation takes place. Here, using the adjacency matrix, every node collects feature information from the neighboring nodes that are connected to it, and from the information gathered, the model learns a more enriched representation by incorporating the context of its surrounding nodes.

a) Layer 1 - First GCN Convolution

$$H^{(1)} = \sigma(\hat{A}XW^{(0)})$$

where:

- $H^{(1)}$ = layer 1 output features
- $\hat{A}$ = normalized adjacency matrix
- X = input features
- $W^{(0)}$ = trainable weight matrix
- σ = activation function (*ReLU*)

Non-Linearity (ReLU Activation):

ReLU activation is applied after the aggregation

$$H^{(1)} = \max(0, H^{(1)})$$

By this we can capture the complex patterns.

Dropout (Regularization, p=0.5):

Drops neurons to reduce overfitting

b) Layer 2 (Output Layer) – Second GCNConv

Second Feature Aggregation: Repeating the process same as Layer1 that gathers features from neighbors.

$$H^{(2)} = \hat{A}H^{(1)}W^{(1)}$$

Classification (Log Softmax Activation): The final result shows class scores of every node

$$Y = \log \text{softmax}(H^{(2)})$$

In our model, the 3 output classes are:

$$0 \rightarrow \text{No Attack}$$

$$1 \rightarrow \text{MITM Attack}$$

$$2 \rightarrow \text{DDoS Attack}$$

The hyperparameters used in our Graph Convolutional Network (GCN) model were carefully selected to ensure optimal training, stability, and generalization. We trained the model for 50 epochs using the Adam optimizer, which adaptively adjusts learning rates based on the first and second moments of gradients to accelerate convergence while reducing oscillations. The initial learning rate was set to 0.01, which balances convergence speed and stability, and L2 regularization (weight decay) of 5×10^{-4} was applied to prevent overfitting by penalizing large weight values. In order to introduce non-linearity and enhance feature extraction, we employed the ReLU activation function in all layers. Additionally, dropout regularization with a probability of 0.5 was incorporated to mitigate overfitting by randomly deactivating a fraction of neurons during training, thereby improving model generalization. However, the weight initialization method was not explicitly specified in the code, meaning default PyTorch initialization techniques were likely used. The architecture was designed to optimize computational efficiency while maintaining robust feature representation. The complete hyperparameter configuration is detailed in Table II.

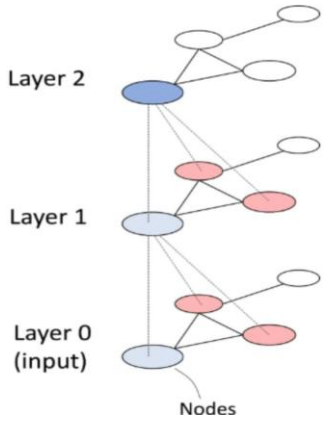

Fig. 3. GCN Layers

TABLE II
HYPERPARAMETER SETTINGS FOR THE GCN MODEL

Hyperparameter	Value
Learning Rate	0.01
Training Epochs	50
Optimizer	Adam
Weight Decay (L2 Regularization)	5×10^{-4}
Activation Function	ReLU
Weight Initialization	Not explicitly set
Dropout Rate	0.5

This part discusses the different techniques and methods adopted to enhance the predictive ability of our Graph Convolutional Network (GCN) model. These methods are centered around optimizing training dynamics, overfitting prevention, and enhancing feature representation. The accurate predictability of our trained model is evident from the generated confusion matrix and it is illustrated in Figure 4. In order to en- hance the predictive performance of our Graph Convolutional Network (GCN) model, we implemented several accuracy-improving techniques. These strategies focused on optimiz- ing training dynamics, preventing overfitting, and improving

feature representation. A learning rate of 0.01 was chosen based on empirical tuning to balance convergence speed and stability. The Adam optimizer, known for its adaptive moment estimation, was employed to dynamically adjust learning rates during training. With the intention of preventing overfitting, L2 regularization (weight decay) of 5×10^{-4} was applied to the model's parameters. This penalizes large weights, ensuring the model generalizes well to unseen data. The model utilizes multiple GCN layers that is layer-wise feature propagation to aggregate information from neighboring nodes efficiently. This hierarchical feature extraction process allows for better representation learning of graph-structured data. Rectified Linear Unit (ReLU) activation was used across all layers to intro- duce non-linearity into the model. ReLU accelerates training convergence by mitigating the vanishing gradient problem. To ensure stable weight distribution and avoid saturation dur- ing training, weight parameters were initialized using Xavier (Glorot) initialization. The model was trained for 50 epochs, with validation loss monitored to ensure optimal stopping. This helps prevent overfitting while allowing the model to reach its best performance. The adjacency matrix was pre- processed with normalization techniques to stabilize training. Proper graph normalization ensures that node representations are efficiently propagated across layers. By integrating these strategies, our model achieves improved accuracy, robustness, and stability across different graph-structured datasets.

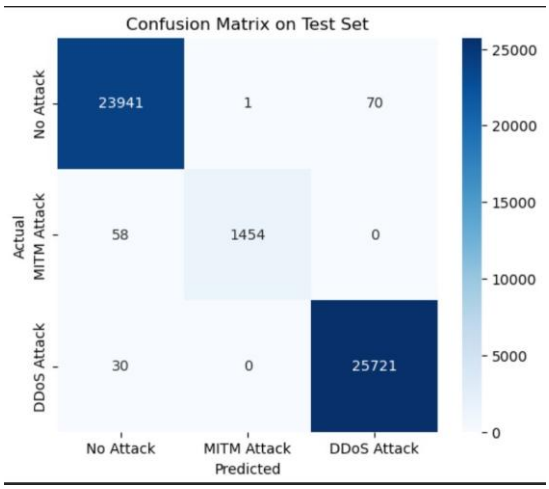

Fig. 4. Confusion Matrix of the trained model.

Our Graph Convolutional Network (GCN) model is designed to classify network traffic into three categories: MITM attack, DDoS attack, and benign (no attack). The classification is performed based on predefined label mappings applied during preprocessing.

During the preprocessing stage, labels are assigned as follows:

- Label 1: Man-in-the-Middle (MITM) attack.
- Label 2: Distributed Denial of Service (DDoS) attack.
- Label 0: No attack (benign traffic).

The trained GCN model takes extracted feature vectors from network traffic as input and produces a probability distribution over the three classes using a softmax activation function. The final predicted class is determined by selecting the class with the highest probability.

For real-time classification, incoming network traffic is processed and transformed into the same feature space as the training dataset. The preprocessed data is then fed into the trained GCN model, which outputs a predicted class label. The classification results can be leveraged for automated intrusion detection and real-time alerting systems, allowing network administrators to respond promptly to potential security threats. In order to classify network traffic in real-time and detect cyber threats, we employed a trained Graph Convolutional Network (GCN) model. The classification process involves feature extraction, model inference, and decision mapping to determine whether the traffic corresponds to a Man-in-the-Middle (MITM) attack, a Distributed Denial-of-Service (DDoS) attack, or benign traffic.

This part elucidates the training and testing procedure of our attack classification model, which is based on a Graph Convolutional Network (GCN). It discusses the important steps involved, such as initialization, the training process, prediction on test data, and testing using different metrics.

The attack classification model is trained and evaluated using a Graph Convolutional Network (GCN). The process consists of four key steps:

1) *Initialization*: Initialize the GCN model parameters (weights and biases). Set hyperparameters such as learning rate, weight decay, and dropout probability.
2) *Training Process*: During the training process, for each epoch e ranging from 1 to the maximum number of epochs (1, max epochs) , the model undergoes a series of steps to optimize its performance. First, a forward pass is executed, where node embeddings are computed using the GCN layers. This is followed by the calculation of the loss, typically using the negative log-likelihood function, which measures how well the model's predictions align with the actual labels. Subsequently, a backward pass is performed to update the model parameters. This is achieved using the Adam optimizer, which adjusts the weights to minimize the loss and improve the model's accuracy over time.
3) *Prediction on Test Data*: Use the trained GCN model to predict attack labels for test data. Assign each node a probability distribution over attack categories using log-softmax activation.
4) *Evaluation*: Evaluate the model using metrics such as accuracy, precision, recall, and F1-score. Generate a confusion matrix to visualize classification performance.

The predicted attack labels are used for real-time classification of network traffic. The classification system enables real-time intrusion detection by continuously analyzing incoming network traffic features. The GCN model captures structural dependencies in the data, improving the robustness and accuracy of the detection mechanism. By leveraging graph-based

learning, the system achieves scalable and efficient threat detection across different network environments.

E. Protecting from DDoS Attacks on a Linux Server

Blocking IP addresses during a DDoS attack can sometimes lead to accidentally blocking legitimate users. To prevent this, instead of outright blocking, we can redirect suspicious traffic to a fake system. This approach wastes the attacker's time and resources without affecting real users. First, set up a fake server that behaves like a real system where malicious activities can be logged. For this, clone the Cowrie repository, set up a Python virtual environment, configure Cowrie, and start the server.To identify suspicious IP addresses with high request rates, use the following command:

```
netstat -ntu | awk '{print $5}' | cut -d: -f1 | sort | uniq -c | sort -nr | head
```

If an IP address makes more than 1000 requests per second, it is flagged as suspicious. Instead of blocking it, we redirect its traffic to the fake server, keeping the real system safe.

F. Encryption and BLE Transmission Algorithm

This segment demonstrates how to encrypt data using AES-CCM (Authenticated Encryption with Associated Data) and send the encrypted data over Bluetooth Low Energy (BLE) to an ESP32 device (supported by ChaCha20, AES-CCM, Ascon, PRESENT, and SPARX encryption algorithms also). The primary steps involved in the encryption *Algorithm 1* are:

Algorithm 1: Generic Encryption Algorithm for Secure BLE Communication

Encryption algorithm choice $A \in$ {AES, ChaCha20, SPARX, PRESENT, ASCON},
Key K, Plaintext P Encrypted packet sent over BLE
Define encryption parameters:
Supported_Algorithms ← {"AES", "ChaCha20", "SPARX", "PRESENT", "ASCON"}
if $A \notin$ *Supported_Algorithms* **then**
    Raise error: "Invalid encryption algorithm selection."
    return
Function `Encrypt(P, K, A):`
switch A **do**
    case *"AES"* **do**
        Generate nonce: $N \leftarrow$ random 12 bytes
        Initialize AES-CCM with key K and nonce N
        $(C, T) \leftarrow$ AES.encrypt_and_digest(P)
        return $N \| C \| T$
    case *"ChaCha20"* **do**
        Generate nonce: $N \leftarrow$ random 12 bytes
        Initialize ChaCha20 with key K and nonce N
        $C \leftarrow$ ChaCha20.encrypt(P)
        return $N \| C$
    case *"SPARX"* **do**
        $C \leftarrow$ SPARX.encrypt(P, K)
        return C
    case *"PRESENT"* **do**
        $C \leftarrow$ PRESENT.encrypt(P, K)
        return C
    case *"ASCON"* **do**
        Generate nonce: $N \leftarrow$ random 16 bytes
        Initialize ASCON-AEAD with key K and nonce N
        $(C, T) \leftarrow$ ASCON.encrypt(P)
        return $N \| C \| T$
Function `Send_Encrypted_Data():`
plaintext ← "Secure Message"
encrypted_data ← `Encrypt(plaintext, K, A)`
Connect to ESP32 using BLE
Send encrypted data via BLE characteristic
Print "Data sent successfully!"
Execute async BLE communication: `Send_Encrypted_Data()`

Algorithm 2: Generic Decryption Algorithm for Secure BLE Communication

```
Encrypted data received via BLE Decrypted plaintext data
Initialize BLE server:
Start BLE service and characteristic with write property
Start BLE advertising
while New encrypted data received do
    Read received data
    Extract algorithm indicator, nonce (if applicable), ciphertext, and authentication tag
    if Valid algorithm indicator then
        Select corresponding decryption method: {"AES", "ChaCha20", "SPARX", "PRESENT", "ASCON"}
    else
        Print "Unknown encryption algorithm"
        return
    Start timer and record memory usage
    Perform decryption:
    Initialize decryption context
    switch Algorithm indicator do
        case "AES" do
            if AES-CCM decryption successful then
                Print decrypted text
            else
                Print "Decryption Failed!"
        case "ChaCha20" do
            Print ChaCha20.decrypt(C)
        case "SPARX" do
            Print SPARX.decrypt(C, K)
        case "PRESENT" do
            Print PRESENT.decrypt(C, K)
        case "ASCON" do
            if ASCON decryption successful then
                Print decrypted text
            else
                Print "Decryption Failed!"
    Record performance metrics and free context
Loop:
while True do
    if New data flag is set then
        Call decryption function and reset flag
    if 5 seconds elapsed then
        Print received packet count and reset timer
```

The function `encrypt_aes_ccm` encrypts a plaintext message using AES with a 128-bit key. A secure 12-byte nonce is generated for each encryption. The encryption mode used is CCM, which combines encryption and authentication (integrity checking). The function then returns a concatenation of the nonce, ciphertext, and authentication tag. Using the `bleak` library, the code asynchronously connects to an ESP32 device for BLE communication using its BLE address and writes the encrypted data to a specific characteristic on the device. The data is transmitted in its encrypted form, ensuring confidentiality and integrity. The encryption and sending of data over BLE are handled in an asynchronous manner, utilizing Python's `asyncio` library to manage the event loop. The data is then decrypted using the appropriate decryption Algorithm 2. The ESP32 uses a BLECharacteristic callback to identify incoming encrypted messages.

The message is structured as follows:

Key	Nonce	Ciphertext	Authentication Tag
1 byte	12 bytes		16 bytes

[1 byte key indicator — 12 bytes nonce — ciphertext — 16 bytes authentication tag]. The first byte of the received message is used to determine the AES key size. The next 12 bytes are extracted as the nonce, and the last 16 bytes contain the authentication tag, which is used to verify whether the message is from a legitimate sender.If the authentication tag is valid, the ciphertext is successfully decrypted, and the plaintext is displayed on the serial monitor along with information such as CPU cycles used, decryption time, and memory consumption. If the authentication fails, the system reports a decryption failure.

IV. IMPLEMENTATION

A. Implementation of a Man-in-the-Middle Attack via GAATTTOL

We used the nRF Connect app on our phone to send a message to an ESP32 device over Bluetooth. During this communication, we performed a Man-in-the-Middle (MITM) attack. A Panda Wireless Bluetooth 4.0 adapter was connected to a Linux laptop, which we used to scan for nearby Bluetooth devices. Using gattool, we connected to our target Bluetooth device by specifying its MAC address. From Figure 5, we can see how we successfully connected to the ESP32 and identified the characteristic UUID used for communication. We explored the device's services and characteristics using the characteristics command. After identifying the relevant UUID, we used it to write a custom message. As a result, the original message sent from the phone was not received by the ESP32; instead, the message injected using atttool/attcack was successfully delivered—demonstrating that our MITM attack effectively intercepted and modified the communication.

```
^Csravanthi@sravanthi-Inspiron-14-7420-2-in-1:~$ sudo gatttool -b A8:42:E3:5B:E7:2A -I
[A8:42:E3:5B:E7:2A][LE]> connect
Attempting to connect to A8:42:E3:5B:E7:2A
Connection successful
[A8:42:E3:5B:E7:2A][LE]> primary
attr handle: 0x0001, end grp handle: 0x0005 uuid: 00001801-0000-1000-8000-00805f9b34fb
attr handle: 0x0014, end grp handle: 0x001c uuid: 00001800-0000-1000-8000-00805f9b34fb
attr handle: 0x0028, end grp handle: 0xffff uuid: abcdef00-1234-5678-1234-56789abcdef0
[A8:42:E3:5B:E7:2A][LE]> characteristics
handle: 0x0002, char properties: 0x20, char value handle: 0x0003, uuid: 00002a05-0000-1000-8000-00805f9b34fb
handle: 0x0015, char properties: 0x02, char value handle: 0x0016, uuid: 00002a00-0000-1000-8000-00805f9b34fb
handle: 0x0017, char properties: 0x02, char value handle: 0x0018, uuid: 00002a01-0000-1000-8000-00805f9b34fb
handle: 0x0019, char properties: 0x02, char value handle: 0x001a, uuid: 00002aa6-0000-1000-8000-00805f9b34fb
handle: 0x0029, char properties: 0x08, char value handle: 0x002a, uuid: abcdef01-1234-5678-1234-56789abcdef0
[A8:42:E3:5B:E7:2A][LE]> char-write-char-write-req 0x002a 48656c6c6f
Characteristic value was written successfully
[A8:42:E3:5B:E7:2A][LE]>
```

Fig. 5. GATT Attack

B. TESTBED SETUP

The Figure 6 provides the visualisation of setting up of our testbed. Using an nRFConnect app on the mobile phone, we sent a message to the ESP32. Here communication is done by BLE Server. The ESP32 listens for incoming data from the phone and message is displayed on OLED screen by using I2C communication. This bluetooth packet is captured in wireshark as shown in Figure 7 and sent to test using trained GCN model. We get what attack is present and if Man-in-the-Middle (MITM) Attack is present, our proposed solution is sending an encrypted message, if DDoS, We create a fake server and send the attacker to fall into that server but the attacker think it is the correct server and waste his time and resources.

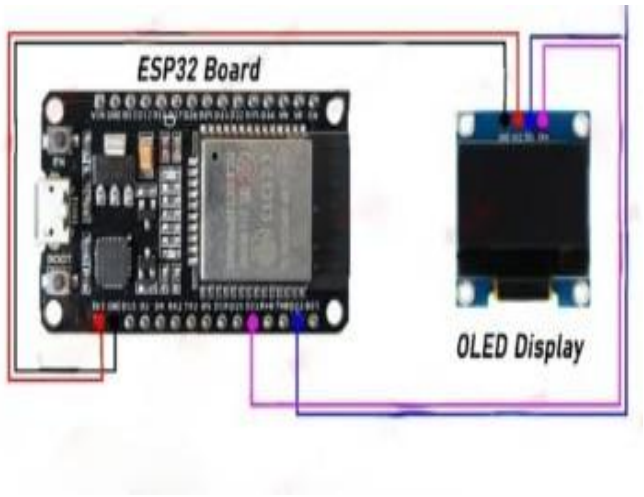

Fig. 6. Testbed Setup: ESP32 Board and OLED Display

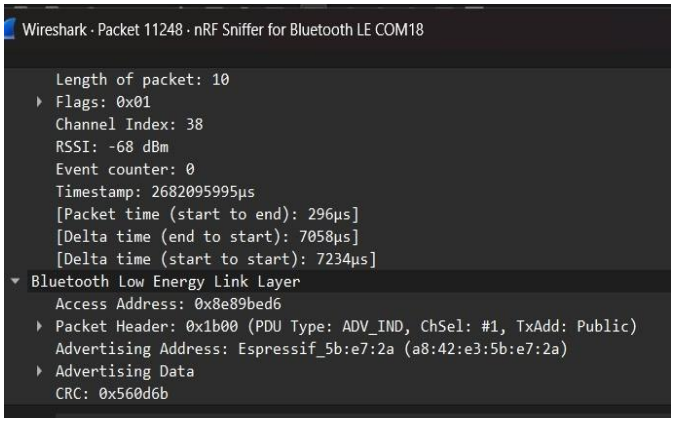

Fig. 7. Wireshark Capture of Bluetooth Low Energy (BLE) Packet

C. RESULTS AND DISCUSSION

This section illustrates the results and discussion of our research findings. It encompasses Ascon selection, which is for safe and effective encryption, where it attributes balanced performance, memory efficiency, strong throughput, and safety. In addition, it elaborates on the performance and effect of GCN on the detection of Bluetooth attacks, placing emphasis on its ability to process graph-structured data, capture global network patterns, and effectively detect MITM and DDOS attacks.

We carried out validation and testing curve analysis to evaluate the model's performance and confirm its resistance to overfitting. The testing and validation accuracy and loss were monitored across 20 epochs. As shown in Figure 8, the validation accuracy steadily increases from about 0.2 to 0.93, while the test accuracy remains stable at around 0.92 throughout the epochs. The small gap between the validation and test accuracy curves demonstrates that the model generalizes well without overfitting. Similarly, from Figure 9, the validation loss declines consistently from approximately 1.6 to 0.15, while the test loss remains nearly constant at around 0.14. This consistent trend indicates that the model performs reliably on unseen data. The use of dropout with a rate of 0.5, which randomly deactivates half of the neurons during training, promotes better generalization and helps prevent overfitting.

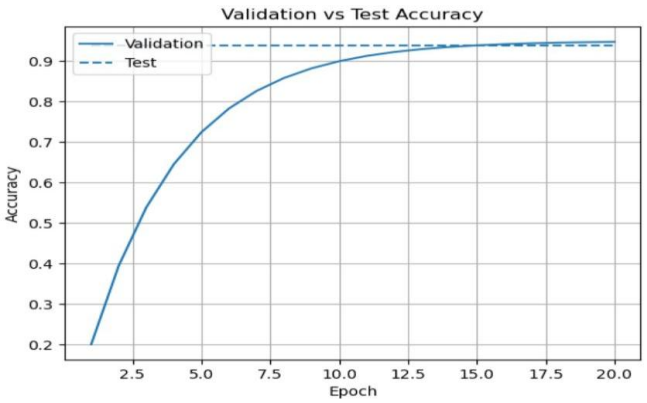

Fig. 8. Testing and Validation Accuracy Curve

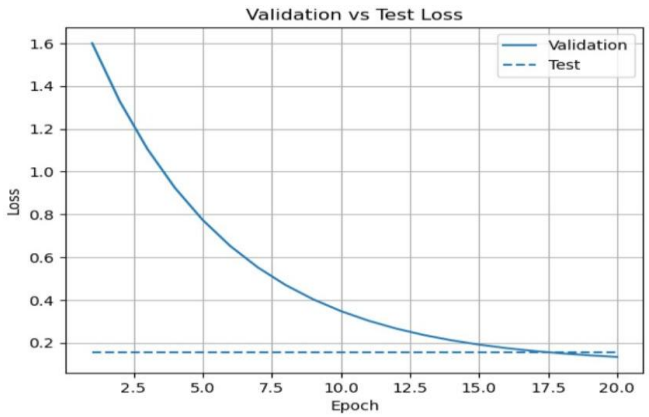

Fig. 9. Testing and Validation Loss Curve

D. Selection of Ascon for Secure and Efficient Encryption

Ascon provides a good trade-off between CPU cycles (50,300) and decryption time (220 µs). It uses fewer CPU cycles than PRESENT (85,600) and SPARX (78,120) while outperforming AES-CCM (269 µs) and SPARX (350 µs). Ascon has balanced performance. Ascon is memory efficient and uses only 580 bytes, less than SPARX (700 bytes) and AES-CCM (624 bytes), making it suitable for IoT and embedded devices that have limited resources. Ascon has a high throughput of 37.24 Mbps, outperforms SPARX (23.40 Mbps) and PRESENT (20.48 Mbps). Ascon is made especially for devices with limitations, providing effective defenses against side-channel attacks. It is secure and lightweight.

Algorithm	Decryption Time (in microseconds)	CPU cycles Used	Memory Used (bytes)	Packets Received	Throughput (Mbps)
AES-CCM	269	63,573	624	1	30.45
ChaCha20-Poly1305	180	42,218	512	1	45.51
SPARX	350	78,120	700	1	23.40
Ascon	220	50,300	580	1	37.24
PRESENT	400	85,600	480	1	20.48

Fig. 10. Comparison of Encryption Algorithms Based on Performance, Resource Efficiency, and Throughput

To evaluate the performance of our models, we employed metrics such as the F measure, accuracy, and area under the ROC (AUC). Accuracy (ACC) represents the ratio of correctly predicted or classified samples to the total number of observations. The measure F provides an evaluation using precision and recall to derive a weighted performance metric between both classes in a binary classification. The area under the ROC curve reflects the ability of the model to distinguish between classes. In Figure 11 True positive (TP) is the number of attack records that are correctly classified as attack. True negative (TN) is the number of benign records correctly classified as benign. False positive (FP) is the number of benign samples incorrectly classified by the classifier as attack. The false negative (FN) is the number of attack samples misclassified as benign.

	Predicted Positive	Predicted Negative	
Actual Positive	TP *True Positive*	FN *False Negative*	Sensitivity $\frac{TP}{(TP+FN)}$
Actual Negative	FP *False Positive*	TN *True Negative*	Specificity $\frac{TN}{(TN+FP)}$
	Precision $\frac{TP}{(TP+FP)}$	Negative Predictive Value $\frac{TN}{(TN+FN)}$	Accuracy $\frac{TP+TN}{(TP+TN+FP+FN)}$

Fig. 11. Confusion Matrix

E. Why Graph Convolutional Networks (GCN)

Graph-Structured Data Handling: Bluetooth communication forms a network in which devices (nodes) interact with each other through edges (connections). Instead of processing the data for each device separately, GCN leads to better generalization and classifies as 'No Attack', 'MITM Attack', and 'DDoS Attack' by learning global network patterns. Bluetooth networks are generally large with many nodes (devices). So, GCN uses localized convolutions on graphs, which requires less memory and

TABLE III
COMPARATIVE ANALYSIS: BLEGUARD VS. PROPOSED GCN MODEL

Aspect	BLEGuard (Cai et al., 2024)	Our GCN Model
Attack Types Handled	BLE Spoofing Attack	Multi-class: MITM, DDoS, No Attack
Model Type	Deep learning + Cyber-physical signal analysis (spectrogram + CNN)	Graph Convolutional Network (GCN)
Input Representation	Time-frequency signal (spectrogram of BLE radio signals)	BLE communication as graph (nodes = devices/messages, edges = sessions/flows)
Feature Engineering	Physical-layer signal analysis + handcrafted predetectors	Graph-structured automatic feature learning
Deployment Target	Smart locks, mobile devices	General BLE-based IoT devices (e.g., wearables, health monitors)
Data Privacy	May require access to raw BLE signals (RF layer)	Works at protocol level doesn't require physical-layer signals
Resilience to Encryption	Robust to encrypted BLE packets	Operates on encrypted session metadata
Classification Granularity	Attack or No Attack	MITM, DDoS, No Attack (more detailed classification)
Performance	96.2% (binary classification)	99.6% (multi-class classification)
Explainability	Low (CNN on spectrograms are hard to interpret)	Medium (GNN explainers can show node/edge contributions)
Lightweight Suitability	Moderate (CNN on edge devices needs optimization)	Lightweight (GNN inference with pruning is edge-deployable)

is more efficient. GCN learns how devices are connected, helping to detect MITM attacks. Regular models analyze data points separately and miss these connections. Figure 12 and Table IV reveal a performance comparisons of the proposed GCN model against the conventional Random Forest classifier. The bar graph in Figure 7 shows how the scores of Random Forest does not meet those of the GCN. Notice how GCN performed better than Random Forest on all of the metrics. The GCN model achieved a 99.7

TABLE IV
COMPARISON BETWEEN GCN AND RANDOM FOREST

Metric	GCN	Random Forest
Accuracy	99.69%	96.54%
F1 Score	98.8%	94.6%
Recall	98.6%	93.7%

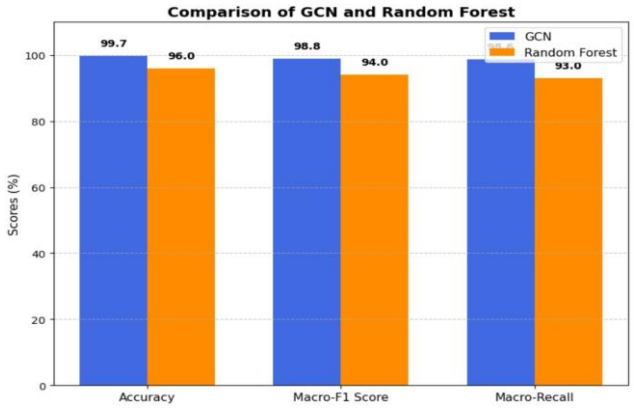

Fig. 12. Comparision of GCN and Random Forest

V. CONCLUSION

In this research,we proposed a complete solution for protecting Bluetooth communication from real-time cyber attacks. Through the combination of lightweight encryption mechanisms and a Graph Convolutional Network (GCN)-based intrusion detection mechanism, we tackled detection and prevention of Man-in-the-Middle (MitM) and Distributed Denial-of-Service (DDoS) attacks.Through our comparative anal- ysis of five lightweight encryption algorithms—ChaCha20, AES-CCM, Ascon, PRESENT, and SPARX—we identified Ascon as the most suitable choice for IoT devices. This conclusion is based on key performance metrics including CPU cycles, decryption time, memory usage, throughput, and adaptability to hardware resource constraints.Our proposed GCN model demonstrated in real-time intrusion detection by successfully classifying Bluetooth communication into benign, MitM, and DDoS attack categories with an impressive 99

Future work includes implementing the proposed Ascon encryption scheme in a real-time Bluetooth communication framework for IoT devices. Verify that the model we proposed is accurate. Additionally, we aim to explore post-quantum lightweight encryption algorithms, such as Kyber and Saber, to future-proof the system against quantum computing threats while considering the resource constraints of IoT devices like the ESP32.

REFERENCES

[1] S. S. Mary, "Bluetooth Security: A Comprehensive Guide to Protecting Your Wireless Devices," Proceedings of the International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), San Diego, CA, USA, Dec. 2024

[2] B. Zhang, "Real-Time Bluetooth Man-in-the-Middle Attack Detection System Based on Graph Neural Network," Communication, Networking and Broadcast Technologies, 2025.

[3] Nthatisi Hlapisi, ”Vulnerabilities and Attacks on Bluetooth LE Devices,” in All About Circuits Technical Reports, 2023.

[4] A. M. Hilal, S. B. H. Hassine, S. Larabi-Marie-Sainte, N. Nemri, M. K. Nour, A. Motwakel, A. S. Zamani, and M. A. Duhayyim, ”Malware Detection Using Decision Tree Based SVM Classifier for IoT,” Comput. Secur., vol. 85, pp. 33–47, 2022

[5] Ray Felch, P. Smith and Q. Brown, ”Machine-in-the-Middle BLE Attack,” Black Hills InfoSec Reports, 2020.

[6] T Melamed, ”MITM Attack on Bluetooth Smart Devices,” in ResearchGate Preprint, 2018.

[7] S. Hızal, A.F.M. S. Akhter, Ü. Ç¸ avus¸og˘lu, and D. Akgu¨n, ”Blockchain-based IoT security solutions for IDS research centers,” Research Article, Sakarya University, Turkey, 2024.

[8] B. A. Alabsi, M. Anbar, and S. D. A. Rihan, ”CNN- CNN: Dual Convolutional Neural Network Approach for Feature Selection and Attack Detection on Internet of Things Networks,” Sensors, vol. 23, no. 6507, 2023

[9] Ahmed Aboukora, Guillaume Bonnet, Florent Galtier, Romain Cayre, Vincent Nicomette, and Guillaume Auriol, ”A Defensive Man-in-the-Middle Approach to Filter BLE Packets,” CEUR Workshop Proceedings, 2021.

[10] M. TajDini, V. Sokolov, and V. Buriachok, “Men-in-the-Middle Attack Simulation on Low Energy Wireless Devices using Software Defined Radio,” Borys Grinchenko Kyiv University, Kyiv, Ukraine.

[11] Ali Mohamamdi Teshnizi, Majid Ghaderi, and Dennis Goeckel, “Covert Communication in Autoencoder Wireless Systems,” in IEEE Transactions on Wireless Communications, vol. 23, no. 5, pp. 847–850, May 2023.

[12] M. Pitchaiah, Philemon Daniel, Praveen, “Implementation of Advanced Encryption Standard Algorithm,” International Journal of Scientific and Engineering Research (IJSER), 2012

[13] X. Che, Y. He, X. Feng, K. Sun, K. Xu, Q. Li, “BlueSWAT: A Lightweight State-Aware Security Framework for Bluetooth Low Energy,” arXiv preprint arXiv:2405.17987, May 2024

[14] Arup Barua, Md Abdullah Al Alamin, Md Shohrab Hossain, and Ekram Hossain, “Security and Privacy Threats for Bluetooth Low Energy in IoT and Wearable Devices: A Comprehensive Survey,” in IEEE Open Journal of the Communications Society, vol. 45, pp. 110–126, 2025

[15] Z. Wang, “Securing Bluetooth Low Energy: A Literature Review,” arXiv preprint arXiv:2404.16846, April 2024.

[16] Tao Liu, Gowri Ramachandran, and Raja Jurdak, “Post-Quantum Cryptography for Internet of Things: A Survey on Performance and Optimization,” in IEEE Transactions on Wireless Communications, vol. 23, no. 5, pp. 847–850, May 2023.

[17] R. La Scala, S. Polese, S. K. Tiwari, A. Visconti, “An Algebraic Attack to the Bluetooth Stream Cipher E0,” arXiv preprint arXiv:2201.01262, Jan. 2022

[18] R. Kaur, “Investigating network traffic using packet sniffing tool-Wireshark,” Int. J. Emerg. Technol. Innov. Res., vol. 6, no. 1, pp. 181–186, 2019.

[19] R. La Scala, S. Polese, S. K. Tiwari, A. Visconti, “An Algebraic Attack to the Bluetooth Stream Cipher E0,” arXiv preprint arXiv:2201.01262, Jan. 2022.

[20] H. Park, C. K. Nkuba, S. Woo, H. Lee, “L2Fuzz: Discovering Bluetooth L2CAP Vulnerabilities Using Stateful Fuzz Testing,” arXiv preprint arXiv:2208.00110, Jul. 2022 .

[21] T. Nagrare, P. Sindhewad, F. Kazi, “BLE Protocol in IoT Devices and Smart Wearable Devices: Security and Privacy Threats,” arXiv preprint arXiv:2301.03852, Jan. 2023 .

[22] D. Antonioli, N. O. Tippenhauer, K. Rasmussen, M. Payer, “BLURtooth: Exploiting Cross-Transport Key Derivation in Bluetooth Classic and Bluetooth Low Energy,” arXiv preprint arXiv:2009.11776, Sep. 2020

[23] A. Cohen, R. G. L. D’Oliveira, K. R. Duffy, J. Woo, M. Me´dard, “AES as Error Correction: Cryp- tosystems for Reliable Communication,” arXiv preprint arXiv:2203.12047, 2022.

[24] K. Doshi, Y. Yilmaz, S. Uludag, “Timely Detection and Mitigation of Stealthy DDoS Attacks via IoT Networks,” arXiv preprint arXiv:2006.08064, Jun. 2020.

[25] A. Barua, M. A. Al Alamin, M. S. Hossain, and E. Hossain, ”Security and Privacy Threats for Bluetooth Low Energy in IoT and Wearable Devices: A Comprehensive Survey,” IEEE Open Journal of the Communications Society, vol. 3, pp. 497-523, Feb. 2022

[26] A. Hekmati, B. Krishnamachari, “Graph-Based DDoS Attack Detection in IoT Systems with Lossy Network,” arXiv preprint arXiv:2403.09118, Mar. 2024.

[27] Ahmad Hani El Fawal, Ali Mansour, Mohammad, “Securing IoT Networks from DDoS Attacks Using a Temporary Dynamic IP Strategy,” PubMed, July 2024.

[28] Misaj Sharafudeen, Vinod Chandra S S, “A Blended Framework for Audio Spoof Detection with Sequential Models and Bags of Auditory Bites,” Scientific Reports, August 2024.

[29] Felipe S Dantas Silva, Esau, “A Taxonomy of DDoS Attack Mitigation Approaches Featured by SDN Technologies in IoT Scenarios,” PubMed, May 2020.

[30] A. Hekmati, N. Jethwa, E. Grippo, B. Krishnamachari, “Correlation-Aware Neural Networks for DDoS Attack Detection in IoT Systems,” arXiv preprint arXiv:2302.07982, Feb. 2023

[31] Sowah Robert A., Ofori-Amanfo Kwadwo B., Mills Godfrey A., Koumadi Koudjo M., “Detection and Prevention of Man-in-the-Middle Spoofing Attacks in MANETs Using Predictive Techniques in Artificial Neural Networks (ANN),” International Journal of Distributed Sensor Networks, Jan. 2019.

[32] Mohammad Najafimehr, Sajjad Zarifzadeh, Seyedakbar Mostafavi, “A Hybrid Machine Learning Approach for Detecting Unprecedented DDoS Attacks,” Journal of Supercomputing, January 2022

[33] X. Che, Y. He, X. Feng, K. Sun, and K. Xu, ”BlueSWAT: A Lightweight State-Aware Security Framework for Bluetooth Low Energy,” Proc. 2024 IEEE Symposium on Security and Privacy (SP), IEEE, 2024.
[34] Xiaoyue Ma, Qiang Zeng, Haotian Chi, Lannan Luo, ”No More Companion Apps Hacking but One Dongle:Hub-Based Blackbox Fuzzing of IoT Firmware”,2023 [Google Scholar]
[35] Rana M. Abdul Haseeb-ur-rehman, Azana Hafizah Mohd Aman, Mohammad Kamrul Hasan,Khairul Akram Zainol Ariffin, Abdallah Namoun, Ali Tufail and Ki-Hyung Kim 4, ”High-Speed Network DDoS Attack Detection: A Survey”,PubMed Central (PMC), Aug. 2023.
[36] Adafruit Industries. (2024). BLE Sniffer with nRF52840
[37] T. Kipf and M. Welling, ”Semi-Supervised Classification with Graph Convolutional Networks,” in Proc. International Conference on Learning Representations (ICLR), Toulon, France, Apr. 2017.
[38] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, ”A Comprehensive Survey on Graph Neural Networks,” IEEE Transactions on Neural Networks and Learning Systems, vol. 32, no. 1, pp. 4–24, 2021.
[39] W. Hamilton, Z. Ying, and J. Leskovec, ”Inductive Representation Learning on Large Graphs,” in Advances in Neural Information Processing Systems (NeurIPS), vol. 30, pp. 1025–1035, 2017.
[40] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, ”Graph Attention Networks,” in Proc. International Conference on Learning Representations (ICLR), Vancouver, Canada, May 2018.
[41] M. Manic and P. Kennedy, ”A Security Analysis for Wireless Sensor Mesh Networks in Highly Critical Systems,” in Proc. IEEE International Symposium on Industrial Electronics (ISIE), Vigo, Spain, Jun. 2007.
[42] C. Bertram, M. L. Campagna, and R. P. Weinmann, ”The Security of ChaCha20-Poly1305 in the Multi-User Setting,” in Proc. IACR Cryptology ePrint Archive, 2023.
[43] L. Beierle, A. Biryukov, J. Großscha¨dl, and L. Perrin, ”Design Strategies for ARX with Provable Bounds: SPARX and LAX,” in Proc. IACR Transactions on Symmetric Cryptology (ToSC), Fukuoka, Japan, Dec. 2016
[44] S. Tillich, G. Banik, and C. Rechberger, ”Security Analysis of ASCON Cipher under Persistent Faults,” in Proc. IACR Cryptology ePrint Archive, 2024.
[45] M. Al-Maadeed, I. Al-Mousa, and M. S. Obaidat, ”Lightweight Cryptography for Internet of Things: A Review,” in Proc. IEEE International Conference on Internet of Things (IoT), Paris, France, Oct. 2022.
[46] G. Shirvani, S. Ghasemshirazi, M. A. Alipour, “Enhancing IoT Security Against DDoS Attacks through Federated Learning,” arXiv preprint arXiv:2403.10968, Mar. 2024.