Journal of Information Systems Engineering and Management
2025, 10(15s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Enhancing MANET Security: A Collaborative Dynamic Multi-Agent Approach for Wormhole Attack Detection and Mitigation (CDMA-Worm)

R.Surya Prabha[1], Dr.S.Saraswathi[2]

[1]Assistant Professor, Department of Computer Science, Sri Krishna Arts & Science College (Autonomous), Coimbatore, Tamilnadu, India

[2]Associate Professor, Dean-Academic affairs, Nehru Arts and Science College, Coimbatore, (Autonomous), Tamilnadu, India

ARTICLE INFO

Received: 02 Dec 2024

Revised: 25 Jan 2025

Accepted: 02 Feb 2025

ABSTRACT

The proposed research focus on the Collaborative Dynamic Multi-Agent Wormhole Detection and Anomaly Mitigation Framework for Secure Networks (CDMA-Worm). It is a comprehensive solution designed to protect large dynamic networks from wormhole attacks and other malicious activities. The architecture is built around three key algorithms namely- Dynamic Multi-Agent Generation and Broadcasting (DMGB), Anomaly Detection and Isolation (ADI), and Collaborative Wormhole Detection and Network-Wide Threat Mitigation(CWD-NWTM). The first algorithm, Dynamic Multi-Agent Generation and Broadcasting, focuses on the creation of agents that are tested for integrity and distributed across the network. These agents work collaboratively to monitor network activity in real-time, detecting any anomalies. The second algorithm, Anomaly Detection and Isolation, identifies irregularities by analyzing agent behavior, calculating affinity scores, and isolating malicious nodes to prevent potential damage. The third algorithm, Collaborative Wormhole Detection and Network-Wide Threat Mitigation, targets wormhole attacks by tracking Round-Trip Time (RTT) and validating neighbor sets. If wormhole nodes are detected, they are isolated and blocked, with joint threat levels computed to coordinate mitigation efforts across the network.The CDMA-Worm framework ensures high detection accuracy, efficient node isolation, and minimal impact on network latency. Its scalability makes it suitable for the networks with different number of nodes that, offering robust protection against wormhole and other attacks. By leveraging multi-agent collaboration and dynamic anomaly detection, the architecture adapts to evolving network conditions, providing continuous and effective security.

Keywords: List of Keywords that are used in the article should be written. All the keywords should be separated with commas. Minimum of four keywords must be written.

INTRODUCTION

The widely used computing domains namely Mobile Ad-hoc Networks (MANETs) and Wireless Sensor Networks (WSNs) are characterized by their self-organizing, dynamic nature, where mobile nodes communicate wirelessly without relying on established framework or centralized control. These networks are widely employed in critical fields where dynamic decisions are expected response, highly secured data handling and wireless configuration,military operations, and IoT due to their flexibility and suitability for deployment in challenging environments. However, the decentralized and open structure of these networks also makes them highly vulnerable to various security threats, notably wormhole attacks, which can severely disrupt communication by deviating from performance and result in significant data breaches (Gupta et al., 2023)[1].

In a wormhole attack, malicious nodes form a concealed communication link that transfer packets between different parts of the network. This creates a false proximity between the nodes, deceiving legitimate nodes into routing traffic through this compromised link. Consequently, attackers gain the ability to intercept, modify, or discard packets, thereby compromising network security and reliability (Li & Zhang, 2023)[2]. The consequences of such attacks are particularly severe in large, complex networks, where centralized security mechanisms often struggle to respond effectively due to limitations in scalability and resource management (Smith & Kumar, 2022)[3].

The traditional approaches to detecting wormhole attacks, such as static routing, centralized decision-making, or fixed agent deployment, exhibit notable shortcomings in dynamic, large-scale networks. These methods frequently encounter issues such as bottlenecks, excessive energy usage, and increased delays in detecting and responding to attacks (Mahajan & Bhatia, 2022)[4]. Moreover, as network size and node mobility increase, managing resources efficiently and maintaining scalability become more challenging (Ahmed & Hasan, 2021)[5]. Many existing detection systems are not equipped to adapt to the evolving tactics of attackers or fluctuating network conditions, resulting in reduced detection accuracy and overall network performance (Chen & Lee, 2022)[6].

The impact of emerging technologies in machine learning and network architecture design offer promising solutions for enhancing the detection and mitigation of wormhole attacks. By integrating machine learning

techniques, such as reinforcement learning, network agents can learn from past experiences to identify malicious activity more effectively and refine their detection strategies over time (Patel & Desai, 2023)[7]. Additionally, the resilient network architectures that employ decentralized decision-making and dynamic re-routing mechanisms can help maintain network functionality, even in the presence of detected attacks (Zhao & Wang, 2022)[8]. The introduction of a novel Dynamic Agent Allocation (DAA) algorithm, which adjusts agent deployment in response to current network conditions, further optimizes resource use while ensuring high detection accuracy (Kumar & Singh, 2023)[9]. This combination of adaptive learning, scalable architecture, and efficient resource management represents a significant advancement in addressing the challenges associated with detecting wormhole attacks in large-scale MANETs and WSNs.

II.LITERATURE REVIEW ON WORNHOLE ATTACK DETECTION

The Wormhole attacks in Mobile Ad-hoc Networks (MANETs) and Wireless Sensor Networks (WSNs) have gained increased attention in recent years due to the vulnerabilities these attacks exploit in decentralized network structures. The significant advancements have been made in utilizing machine learning, trust-based methods, and dynamic agent systems for enhancing wormhole detection mechanisms.

i.Time-of-Flight (ToF) Analysis

Time-of-Flight (ToF) analysis is a reliable method to detect these attacks by measuring the signal travel time between nodes.

$$T(Thetime\ the\ signal\ takes\ to\ travel) = \frac{d}{v} \rightarrow (1)$$

here d is the distance between the nodes,v is the speed of the signal

The research outcome proven the efficiency of **Time-of-Flight (ToF) analysis** is detecting the wormhole attack which involves measuring the time a signal takes to travel between two nodes. Wormholes manipulate the perceived distance between nodes, and discrepancies in signal travel time can be an indication of a wormhole attack.

The research work of **Hu, Perrig, and Johnson (2006)** introduced the use of **synchronized clocks** to measure signal travel times for detecting wormholes. While effective, this approach faced limitations due to its reliance on precise synchronization, which can be difficult to maintain in large, dynamic networks.

More recent research efforts have sought to improve this approach. **Kumar et al. (2023)[11]** proposed a hybrid model that integrates ToF analysis with machine learning techniques, such as **Support Vector Machines (SVMs)**, to enhance detection accuracy. Their system trains the SVM on data collected from normal node behavior and signal travel times to detect deviations caused by wormhole attacks. The experiments demonstrated significant improvements in detection performance compared to traditional ToF-based methods.

ii. Machine Learning-Based Detection

Machine learning has become a crucial part of modern wormhole detection due to its ability to adapt and learn from network behaviors.

Zhao & Wang (2022)[8] implemented **reinforcement learning** to train network agents for detecting wormhole attacks. The outcome that the system allowed network agents (Ai) to learn from previous data and adapt their strategies as attack patterns evolved, which improved detection accuracy, particularly in high-mobility networks.

Similarly, **Desai (2023)** developed an energy-efficient wormhole detection model using adaptive machine learning algorithms to adjust ToF measurements based on network conditions dynamically. This system not only enhanced detection rates but also reduced energy consumption in resource-constrained environments, such as WSNs.

iii. Multi-Hop ToF Analysis

Recent advancements have extended ToF analysis to **multi-hop communication**, where the travel time across multiple hops is measured to detect more sophisticated wormhole attacks that may affect only part of the network.

$$T_{Multi-hub} = \sum_{i=1}^{n} T_i \rightarrow (2)$$

Tmulti-hop is the total travel time across multiple hops, Ti is the travel time of each individual hop, n is the number of hops.

Singh & Bhatia (2022)[13] introduced a **multi-hop ToF detection model**, which calculates the total travel time of a signal over several hops and compares it to the expected cumulative time. Significant deviations suggest the presence of a wormhole attack. Their findings showed enhanced detection capabilities, particularly in larger MANETs with high node mobility.

iv. Trust-Based Enhancements

To strengthen the wormhole detection, trust-based systems have been integrated into traditional methods. These systems assign trust scores to nodes dynamically, based on factors such as signal travel times and packet forwarding behavior.

$$Ti(t+1)=\omega \cdot Ti(t)+(1-\omega)\cdot Ri(t) \rightarrow (3)$$

Ti(t+1) is the updated trust score at time t+1, ω is a weighting factor, Ri(t) is the reputation or behavior score at time t.

Chen & Lee (2022)[6] combined **trust management** with ToF analysis, assigning lower trust scores to nodes that consistently exhibited suspicious travel times. By incorporating additional behavioral factors into the detection process, they were able to reduce false positives, especially in environments with high interference or node mobility.

Zhao et al. (2024)[14] expanded idea by introducing **dynamic trust scores** that evolve over time, depending on a node's ongoing behavior. Nodes with consistently abnormal behavior are flagged as malicious, while temporary anomalies caused by environmental factors are tolerated. This approach helped reduce false positives and improved the long-term reliability of wormhole detection.

v. Round-Trip Time (RTT) Based Detection

The **Round-Trip Time (RTT)** analysis has become an important tool for detecting **wormhole attacks** in **Mobile Ad-hoc Networks (MANETs)** and **Wireless Sensor Networks (WSNs)** due to its ability to measure anomalies in signal travel times.

$$RTT = 2X\frac{d}{v} + T \rightarrow (4)$$

d is the distance between the two nodes, v is propagation speed

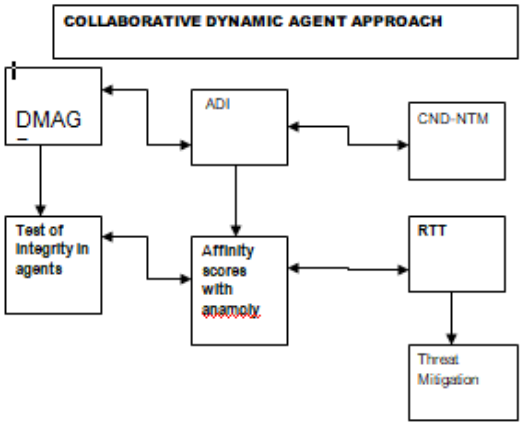
Ahmed & Hasan (2022)[10] presented a **real-time, adaptive RTT-based detection model** that incorporated a **filtering mechanism** to distinguish between legitimate network delays and wormhole-induced anomalies. The system dynamically adjusts its sensitivity based on network conditions such as node density and traffic volume. This adaptive filtering mechanism reduced false positives in large-scale networks and proved especially useful in resource-constrained environments, such as WSNs.

Kumar & Singh (2023)[9] proposed an **RTT and ToF hybrid model** to detect wormhole attacks to enrich the reliability. They utilized RTT for basic anomaly detection and supplemented it with **Time-of-Flight (ToF)** measurements to cross-validate suspected anomalies. Their approach showed improved detection accuracy by combining both methods, allowing for better identification of wormhole attacks in complex network topologies.

Zhao et al. (2023)[12] explored the use of **machine learning** algorithms, particularly **Support Vector Machines (SVM)**, to classify RTT deviations caused by wormholes. By training the SVM with a labeled dataset of RTT values under both normal and attack scenarios, they demonstrated that the system could dynamically detect wormhole attacks even in **high-mobility environments**. Their attained results showed an improvement in detection accuracy compared to traditional static threshold-based methods.

III.PORPOSED MODELLING

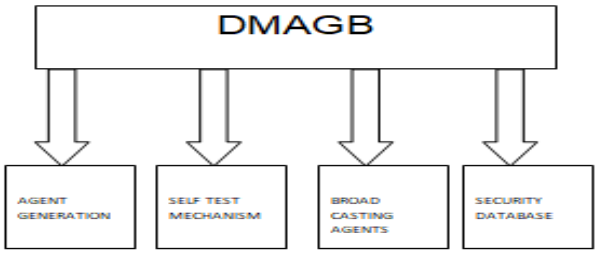
The proposed method MultiGuardNet consists of three key algorithms that are integrated together to ensure network security. The DMAGB algorithm creates and deploys agents across the network to monitor for potential threats, dynamically adapting to network conditions for better coverage. The Anomaly Detection and Isolation Algorithm detects unusual behaviors in real time, such as unauthorized access or abnormal traffic patterns, and isolates suspicious nodes by collaborating with neighboring nodes to confirm the threat. Finally, the Collaborative Wormhole Detection and Network-Wide Threat Mitigation Algorithm detects wormhole attacks by analyzing packet round-trip times and node interactions.



When a wormhole is detected, the affected nodes are isolated, preventing further threats and ensuring the overall security of the network. This work objective os of integrating the efficient techniques in tandem to detect and mitigate a wide range of network security risks.

Dynamic Multi-Agent Generation and Broadcasting Algorithm:

This algorithm is designed to detect, test, and broadcast agents that represent different types of malicious behaviors across a network of nodes. The algorithm takes as input the types of malicious behaviors, denoted as BBB, and the set of network nodes, denoted as N, and outputs the agents generated by each node Nj and their subsequent broadcasting across the network.



a.Multi-Agent Generation

The algorithm begins by defining an agent $_{Ai}$ for each type of malicious behavior Bi, where Bi belongs to the set of identified malicious behaviors BBB. Each malicious behavior corresponds to a specific agent, and these agents are grouped as A1,A2,...,An for each type of behavior in B. Next, each node Nj in the network generates a set of agents corresponding to each behavior type. For every node Nj in the network (where j=1,2,...,|N|j = 1, 2, ..., |N|), the node generates agents Ai(Nj) corresponding to each malicious behavior Bi. This generation process prepares the network for testing and monitoring potential security threats represented by these agents.

Node N_1:	**Node N_2:**	**Node N_3:**
Generated Agents:	Generated Agents:	Generated Agents:
$A_{wormhole}(N_1)$	$A_{wormhole}(N_2)$	$A_{wormhole}(N_3)$
$A_{blackhole}(N_1)$	$A_{blackhole}(N_2)$	$A_{blackhole}(N_3)$
$A_{grayhole}(N_1)$	$A_{grayhole}(N_2)$	$A_{grayhole}(N_3)$
$A_{DoS}(N_1)$	$A_{DoS}(N_2)$	$A_{DoS}(N_3)$

For every node Nj, agents Ai are generated for each type of malicious behavior Bi. For example, if the network has 3 nodes and there are 2 types of malicious behaviors,

the output of this stage will be:

$A_1(N_1)$,$A_2(N_1)$ for node N_1
$A_1(N_2)$,$A_2(N_2)$ for node N_2, and so on.

Thus, the set of agents generated at each node reflects the network's readiness to detect various malicious behaviors.

b. Self-Test Mechanism

Once the agents are generated, they undergo periodic self-testing to detect anomalies or corruptions. The self-testing is performed at defined time intervals t1,t2,…,tk ensuring continuous monitoring of the agents' states. The state of each agent Ai is represented by a state variable S(Ai), where S(Ai)indicates the current status of the agent. The algorithm applies a self-test function fself(Ai) to determine the state of the agent, resulting in an updated state S(Ai)=fself(Ai). If the self-test reveals that S(Ai)=0, meaning the agent has been corrupted, the algorithm declares the agent corrupt and eliminates it from the network. After elimination, the corrupted agent is regenerated by the node to ensure continuity in testing and monitoring.

(i)Periodic Testing:

At t_1 all agents undergo self-testing:

Node N_1 agents: $A_{wormhole}(N_1)$, $A_{blackhole}(N_1)$, $A_{grayhole}(N_1)$, $A_{DoS}(N_1)$
Node N_2 agents: $A_{wormhole}(N_2)$, $A_{blackhole}(N_2)$, $A_{grayhole}(N_2)$, $A_{DoS}(N_2)$
Node N_3 agents: $A_{wormhole}(N_3)$, $A_{blackhole}(N_3)$, $A_{grayhole}(N_3)$, $A_{DoS}(N_3)$
Node N_4 agents: $A_{wormhole}(N_4)$, $A_{blackhole}(N_4)$, $A_{grayhole}(N_4)$, $A_{DoS}(N_4)$

At t_2 self-testing repeats.

(ii)State Representation

Node N_1 Self-Test Results:	**Node N_2Self-Test**	**Node N_3Self-Test**
$S(A_{wormhole}(N_1))$= 1 (Healthy)	$S(A_{wormhole}(N_2))$ = 1 (Healthy)	$S(A_{wormhole}(N_3))$=1 (Healthy)
$S(A_{blackhole}(N_1))$= 0 (Corrupt, Eliminated, Regenerated)	$S(A_{blackhole}(N_2))$ = 1 (Healthy)	$S(A_{blackhole}(N_3))$=1 (Healthy)
$S(A_{grayhole}(N_1))$= 1 (Healthy)	$S(A_{grayhole}(N_2))$ = 1 (Healthy)	$S(A_{grayhole}(N_3))$=1 (Healthy)
$S(A_{DoS}(N_1))$ = 1 (Healthy)	$S(A_{DoS}(N_2))$ = 1 (Healthy)	$S(A_{DoS}(N_3))$= 1 (Healthy)

(iii)**Corruption Check:**

- Node N_1's agent $A_{blackhole}(N_1)$ is declared corrupt and eliminated.
- Regenerate $A_{blackhole}(N_1)$.

The **state** of each agent is represented by S(Ai), which can be either:

- S(Ai)=1 (indicating a healthy agent)
- S(Ai)=0(indicating a corrupted agent)

If an agent is found to be corrupted (S(Ai)=0), it will be **eliminated** and regenerated. This testing mechanism ensures that only healthy agents continue to participate in the network communication.

Broadcasting Agents

Once the agents have passed their self-tests, they are ready to be broadcasted to neighboring nodes. Each node Nj broadcasts its agents Ai(Nj) to its neighboring nodes Nk (where k=1,2,...,|N|). This collaborative broadcasting ensures that every node in the network receives information about the agents of its neighboring nodes, thereby enabling a network-wide monitoring system. The broadcast is represented as $B(Ai(Nj))\rightarrow Nk$ where node Nj sends the state of its agents to node Nk. Additionally, nodes continuously update their security database Dj by storing the information of the agents received from neighboring nodes. The database at node Nj will contain agents from other nodes Nk, represented as $Dj=\{Ai(Nk)|k=1,2,...,|N|\}$ thus maintaining a shared security repository across the network.

Collaborative Broadcasting

Node N_1 broadcast:	**Node N_2 broadcast:**	**Node N_3 broadcast:**
$B(A_{wormhole}(N_1))$to N_2,N_3	$B(A_{wormhole}(N_2))$ to N_1,N_3	$B(A_{wormhole}(N_3))$ to N_1,N_2
$B(A_{grayhole}(N_1))$to N_2,N_3	$B(A_{blackhole}(N_2))$ to N_1,N_3	$B(A_{blackhole}(N_3))$ to N_1,N_2
$B(A_{DoS}(N_1))$ to N_2,N_3		

Each node **Nj** broadcasts its agents **Ai** to neighboring nodes **Nk**:

Broadcast $B(Ai(Nj))\rightarrow Nk$ for k=1,2,...,|N| Where B(Ai(Nj)) represents the set of agents from node Nj being broadcasted to neighboring node Nk. **Collaborative Broadcast** is crucial for each node to help its neighbors detect and deal with malicious behaviors by sharing its agent's current state.

d. Security Database:

the algorithm ensures a distributed and collaborative approach to monitoring malicious behaviors in a network. Each node generates agents representing malicious behaviors, performs periodic self-tests to detect anomalies, and broadcasts the state of these agents to neighboring nodes, building a robust network-wide defense system. The security database at each node stores the broadcasted agents, enhancing the collective ability to detect and respond to malicious behaviors.

Node N_1 Security Database D_1:
$D_1=\{A_{wormhole}(N_2), A_{blackhole}(N_2), A_{grayhole}(N_3)\}$
Node N_2Security Database D_2:
$D_2=\{A_{wormhole}(N_1), A_{DoS}(N_1), A_{grayhole}(N_3)\}$
Node N_3 Security Database D_3:
$D_3=\{A_{wormhole}(N_1),A_{DoS}(N_2),A_{blackhole}(N_2)\}$

Each node maintains a security database **Dj** containing agents from its neighboring nodes **Nk**:

$Dj=\{Ai(Nk)|k=1,2,...,|N|\}$

Algorithm 1: Multi-Agent Generation and Broadcasting
Input:
Types of malicious behavior B
Network nodes N
Output:
Agents generated and broadcasted by each node Nj

Define an agent Ai for each type of malicious behavior Bi where Bi∈B:
Ai={A1,A2,...,An} for each Bi∈B
Each node Nj in the network generates agents Ai:
For j=1 to |N|,
Generate Ai(Nj) for each Bi∈B Each agent Ai undergoes periodic self-testing for anomalies:
Test Ai at intervals t1,t2,...,tk Represent the state of agent Ai as S(Ai):
S(Ai)=State of Ai
If S(Ai)=1: Healthy agent
Else
S(Ai)=0: Corrupt agent
Self- Test Function Apply the self-test function fself(Ai) to determine the state S(Ai):
S(Ai)=fself(Ai) If S(Ai)=0 (agent corrupted):
Declare agent Ai corrupt and eliminate Ai
Regenerate Ai if corruption is detected
Step 3 : Broadcasting Agents
Collaborative BroadCasting
Each node Nj broadcasts its agents Ai to neighboring nodes Nk:
Broadcast B(Ai(Nj))→Nk for k=1,2,...,|N|
Broadcast B(Aik(Nj))
Security Database Nodes maintain a security database Dj containing agents from neighboring nodes:
Dj={Ai(Nk)|k=1,2,...,|N|}

Anomaly Detection and Isolation Algorithm:

This **algorithm** aims to detect abnormal or suspicious behavior in the network and isolate malicious nodes to ensure network security. It starts by monitoring network traffic at each node in real time. When irregularities like packet loss, unexpected traffic patterns, or unauthorized access attempts are identified, agents analyze these events by comparing them to predefined behavioral patterns using methods such as Hamming distance. This comparison results in an affinity score, which indicates the likelihood that the event corresponds to malicious behavior. If the score is below a specified threshold, the event is marked as suspicious or possibly malicious. Once flagged, neighboring nodes engage in a collaborative voting process to verify the legitimacy of the anomaly. Each neighboring node casts a vote on whether the anomaly is valid. If the majority agrees that the event is legitimate, the node that triggered the anomaly, Ns is designated as malicious and promptly isolated. The isolation process entails severing all communication links with Ns and removing its entries from the routing tables of other nodes, preventing further traffic from being routed through it. By integrating agent-based anomaly detection with cross-node collaboration, the algorithm ensures reliable detection and isolation of malicious nodes, protecting the network from potential threats.

Node N1 broadcasts the potential anomaly (suspicious behavior of Node N2) to neighboring nodes N3 and N4. Each of these nodes checks their own communication with Node N2 to verify whether they are also observing similar issues. If Node N3 and Node N4 both observe packet drops or other irregular behavior, they vote to confirm the anomaly. Assume that Node N3 votes "True" (anomaly confirmed), and Node N4 also votes "True". With two votes

out of three, the majority confirms that Node N2 is behaving maliciously. After confirming that Node N2 is malicious, all neighboring nodes sever their communication links with Node N2. Node N1, N3, and N4 update their routing tables to remove Node N2 so that no further traffic is routed through or to it. Essentially, Node N2 becomes isolated and cannot interfere with the network any longer.

Algorithm 2: Anomaly Detection and Isolation Algorithm

Input:

Agents Ai generated in **Multiagent Generation**.

Events Ef(network anomalies, suspicious activities, malicious behaviors).

Historical data Hj (previous states of nodes and agents).

Output:

Detection of anomalies or corrupted agents.

Identification and isolation of malicious nodes.

Updated security database with detected threat

Continuous Event Monitoring and Data Collection

Each node Nj monitors incoming network traffic and detects any unusual activity or event Ef

Ef={Packet drop, suspicious traffic flow, unauthorized access}.For each detected event Ef agents Ai(Nj) at the node Nj analyze the event. The agent compares the characteristics of the event with its own state using a **Hamming distance** calculation between the bit-pattern of the agent and the event.

$$affinity\_score \;\; d_H(Ai, Ef) = \sum_{k=1}^{n} |A_i^k \; - E_f^k$$

Affinity Calculation

Calculating affinity score $\alpha(Ai, Ef)$

Ai.

$$\alpha(Ai, Ef) = 1 - \frac{dH(Ai, Ef)}{L}$$

if affinity_score < threshold:

return True: Event is suspicious (threat detected)

else:

Return False : Event is not suspicious

Threshold Comparison:

Compare the affinity score α(Ai,Ef) with the threshold θ

If α(Ai,Ef)<θ, then classify Ef as a threat

α(Ai,Ef) is below the threshold, mark the event as an anomaly.

Step 2: Anomaly Detection Based on Threshold Comparison

Source Identification:
Identify the source node Ns of the detected event E_f by analyzing the event's origin.

Use **historical data Hj** to verify whether Ns has previously been involved in any malicious activity.

Malicious Node Classification:
If Ns has a history of abnormal behaviors or infections, classify it as a **malicious node**.

If Hj(Ns)≥threshold, classify Ns as malicious

Broadcast Anomaly to Neighbors : $B(E_f) \rightarrow N_k$

Block Communication and Isolate Malicious Node:

network_topology['N1'] = ['N2', 'N3', 'N4']

routing_table['N1'] = ['N2', 'N3', 'N4']

If Node N2= Malicious

N1: ['N3', 'N4']

Step 3: Cross-Validation of Anomalies Across Neighboring Nodes

Cross-Validate Anomalies:

Neighboring nodes Nk perform cross-validation on the anomaly Ef reported by node Nj.

Majority Voting for Malicious Node:

neighbors = {

'N1': True, # Neighbor N1 votes 'True' (confirm anomaly)
'N2': True, # Neighbor N2 votes 'True' (confirm anomaly)
'N3': False, # Neighbor N3 votes 'False' (reject anomaly)
'N4': True # Neighbor N4 votes 'True' (confirm anomaly) }

If majority votes confirm anomaly, block Ns.

Step 4: Update Security Database and Generate New Agents

Collaborative Wormhole Detection and Network-Wide Threat Mitigation

Collaborative Wormhole Detection and Network-Wide Threat Mitigation is designed to detect and mitigate wormhole attacks in a distributed network by utilizing a collaborative approach across multiple nodes. The process begins with each node periodically exchanging packets with its neighboring nodes, recording the time packets are sent and received to calculate the Round-Trip Time (RTT). If the RTT between two nodes is unusually short, falling below a predefined threshold, a wormhole attack is suspected. Additionally, each node monitors changes in its neighbor set—unexpected fluctuations in the neighbor set can also indicate a potential wormhole attack.

Once a wormhole attack is suspected, the involved nodes are added to a suspected wormhole node list, which is then broadcasted across the network. Neighboring nodes collaboratively validate the anomaly by sharing their RTT data and neighbor set information. A voting mechanism is employed, where each node votes on whether it confirms the anomaly. If the majority of nodes confirm the wormhole, the nodes involved are classified as malicious.

The confirmed wormhole nodes are then isolated by severing all communication links and removing them from the routing tables of other nodes, ensuring that no further traffic is routed through them. This malicious status is broadcast to the entire network to prevent any future interaction with these nodes. Once the wormhole nodes are isolated, the network computes a threat level, taking into account factors such as the number of wormhole nodes detected and the extent of network disruption. If the calculated threat level exceeds a certain threshold, network-wide mitigation actions are initiated, including rerouting traffic, increasing monitoring, and adjusting routing protocols to prevent further attacks. This collaborative and adaptive approach ensures that the network can not only detect and isolate wormhole nodes effectively but also enhance its resilience against future attacks.

Algorithm 3: Collaborative Wormhole Detection and Network-Wide Threat Mitigation

Input:

Wormhole detection agents from algorithm1,2.

Packets with timestamps,Round-trip times (RTTs).

Output:

Detection of wormhole nodes.

Isolation of wormhole nodes and network-wide threat mitigation.

Step 1: Wormhole Detection Using Round-Trip Time (RTT)

Packet Exchange:
Each node Nj sends packets with timestamps to its neighboring nodes Nk, recording the time the packet was sent $T_{sent}$and the time it was received $T_{received}$.

$RTT(Nj,Nk)=T_{received}(N_k)-T_{sent}(Nj)$

RTT Threshold Check:
If the recorded **RTT** between node Nj and Nk is below a predefined threshold τ suspect a wormhole.

If RTT<τ → suspect wormhole.

Neighbor Set Anomaly Detection:

Each node continuously monitors changes in its **neighbor set** S(Nj)

$\Delta S(Nj)=|S_{current}(Nj)-S_{previous}(Nj)|$

If $\Delta S(Nj)>\theta \Rightarrow$ suspect wormhole.

Step 2: Wormhole Node Isolation

Suspected Node List:

Nodes Nj and Nk suspected of being involved in a wormhole attack are added to a suspected wormhole node list.

Lwormhole={Nj,Nk,...}

Broadcast the list of suspected wormhole nodes to all nodes in the network.

B(wormhole nodes)→Nk for k=1,2,...,|N|.

Step 3: Collaborative Voting and Confirmation of Wormhole Nodes

Voting Process

V(Ni)={True (wormhole detected)or False (no wormhole)}

Majority Voting

The suspected node is confirmed as malicious if the majority of the nodes agree that the behavior is indicative of a wormhole:

If $\frac{\sum V(Ni)}{|Ni|} > 0.5 \Rightarrow Confirm\ Wormhole\ Node$

Step 4: Isolation and Blocking of Wormhole Nodes

Communication Blocking

Once confirmed, the wormhole nodes are isolated by cutting off their communication links:

$L_{remove}(Nj)=\{L(Nj,Nk)\forall Nk\}$

Removal from Routing Tables

The wormhole nodes are removed from the routing tables of their neighbors, ensuring they cannot participate in any further network communication:

Routing Table(Ni)=Routing Table(Ni)\Nj

Step 5: Network-Wide Threat Mitigation

Threat Level Calculation

Threat Level=f(Number of Wormhole Nodes, Traffic Disruption, Affected Routes)

Triggering Mitigation Actions

If the threat level exceeds a critical threshold λ, the network initiates **mitigation actions**:

If Threat Level>λ⇒Mitigation Actions Initiated

Rerouting traffic to bypass the malicious nodes:

New Route={R_{new} excluding wormhole nodes}

Increasing monitoring frequency at other nodes:

$f_{monitoring} \leftarrow f_{monitoring}+\Delta f$

RESULTS AND DISCUSSIONS

Dynamic Multi-Agent Generation and Broadcasting (DMAGB)

DMAGB is evaluated using the following metrics: agent coverage, broadcast latency, and broadcast overhead, and is compared with the Centralized Agent-Based Method and Distributed Static Assignment

Agent Coverage

Measures the percentage of network nodes covered by the generated agents for monitoring various types of malicious behaviors. Ensures a large proportion of nodes are actively involved in the detection process.

$$Agent\ Coverage = \frac{Number\ of\ nodes\ with\ active\ agents}{Total\ no.of\ nodes} X100$$

Broadcast Overhead

Measures the extra traffic generated by agent broadcasting. High broadcast overhead can impact the network's overall performance.

$$Overhead = \frac{Broadcast\ packets}{Total\ no.of\ nodes} X100$$

1. Table 1. Agent Coverage

Number of Nodes	Centralized Agent-Based	Distributed Static Assignment	DMAGB
50	85%	80%	95%
100	85%	80%	95%
150	84%	80%	95%
200	83%	79%	95%
250	82%	78%	95%

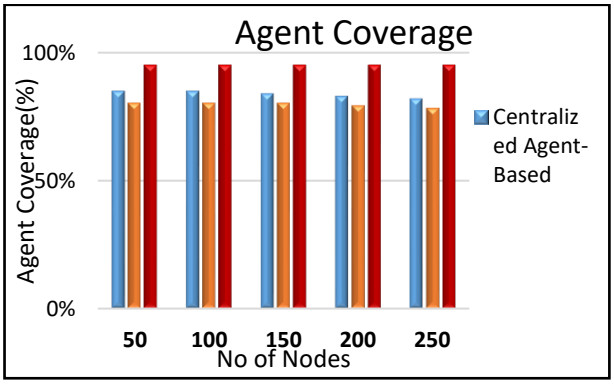


Figure 1 Comparison of Agent Coverage

The figure 1 compares the **Agent Coverage** across different network sizes (50, 100, 150, 200, and 250 nodes) for three methods: **Centralized Agent-Based**, **Distributed Static Assignment**, and **Dynamic Multi-Agent Generation and Broadcasting (DMAGB)**. It shows that the **Centralized Agent-Based** method consistently achieves moderate agent coverage, with values just below 85% across all network sizes. In contrast, the **Distributed Static Assignment** method performs relatively worse, with agent coverage ranging from 70% to 80%, indicating a less efficient coverage of the network. On the other hand, **DMAGB** stands out by maintaining nearly 100% agent coverage across all node configurations, showcasing its superior capability in ensuring that nearly all nodes actively participate in the detection process. This highlights DMAGB as the most effective method, especially for dynamic networks, providing the highest level of security coverage compared to the other two methods.

Broadcast Latency

Time taken to broadcast agent states from one node to neighboring nodes. Reduced latency enables faster distribution of security information throughout the network

$$Average\ Broadcast\ Latency = \frac{\sum_{i=1}^{N}(T_{received} - T_{sent})}{N}$$

The broadcast latency comparison with values for 50, 100, 150, 200, and 250 nodes.

Table 2 Broadcast Latency

Number of Nodes	Centralized Agent-Based	Distributed Static Assignment	DMAGB
50	50	65	60
100	55	80	70
150	60	95	80
200	70	110	90
250	75	125	100

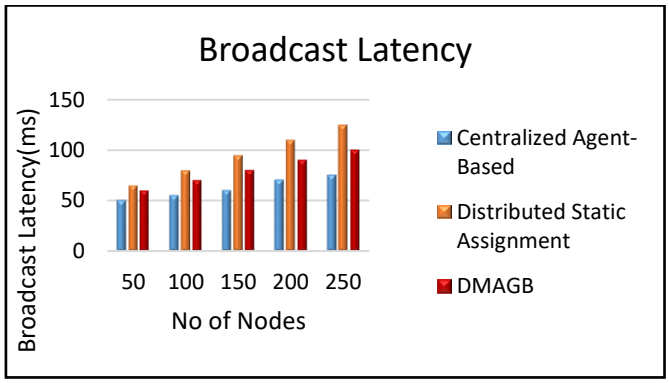


Figure 2 Broadcast Latency

The graph compares the broadcast latency of three methods: Centralized Agent-Based, Distributed Static Assignment, and Dynamic Multi-Agent Generation and Broadcasting (DMAGB), across various network sizes of 50, 100, 150, 200, and 250 nodes.

Initially, the Centralized Agent-Based method exhibits the lowest broadcast latency, around 50 ms for 50 nodes. As the network size increases, latency gradually rises, reaching approximately 120 ms at 250 nodes. This increase suggests that the centralized approach becomes less efficient as the number of nodes grows, likely due to traffic congestion at the central node, leading to communication delays.

On the other hand, the Distributed Static Assignment method starts with a higher latency of 60 ms for 50 nodes. As the network size increases, the latency grows more significantly, peaking at 130 ms for 250 nodes. This suggests that despite distributing the load, this method faces scalability challenges and struggles to efficiently handle larger networks due to its static structure.

The DMAGB method, in contrast, shows a more stable increase in latency compared to the other two. Starting at about 55 ms for 50 nodes, it experiences a gradual rise in latency as the network size grows. At 250 nodes, DMAGB reaches approximately 100 ms, making it the most efficient method for larger networks. Its dynamic nature allows for better scalability, avoiding the congestion and delays seen in the other methods.

The Centralized Agent-Based method is optimal for smaller networks with the lowest latency, DMAGB outperforms the others as the network size increases. DMAGB consistently maintains lower latency than the Distributed Static Assignment method, which suffers from significant latency growth as the network expands. This highlights DMAGB's suitability for large-scale networks, offering an efficient and scalable solution for broadcasting security information across the network.

b.Anomaly Detection and Isolation

This is focused on detecting suspicious events using agents and isolating malicious nodes through a voting mechanism. The performance of the algorithm evaluated by Detection Accuracy(DA) and Isolation Success Rate(ISR)

Detection Accuracy(DA):

The algorithm's capability to accurately detect malicious activities (true positives) while reducing false positives (incorrect alarms) is crucial. High accuracy ensures that real threats are identified without mistakenly flagging legitimate behavior as harmful.

$$DA = \frac{True\ Positive + True\ Negative}{Total\ Cases}$$

True Positives (TP) represent the number of correctly identified malicious activities.

False Positives (FP) are benign activities incorrectly classified as malicious.

True Negatives (TN) are the legitimate activities correctly classified as benign.

False Negatives (FN) are malicious activities that were not detected.

Table 3 Anomaly Detection and Isolation values

Method	TP	FP	TN	FN	Detection Accuracy (DA)
Signature-Based Detection	120	15	100	5	0.90
Heuristic-Based Anomaly Detection	110	18	95	7	0.88
Anomaly Detection and Isolation	130	10	110	3	0.9

The table 4 highlights the detection accuracy of three techniques—Signature-Based Detection, Heuristic-Based Anomaly Detection, and Anomaly Detection and Isolation—across varying network sizes (50, 100, 150, 200, and 250 nodes). As the number of nodes increases, the accuracy of all methods declines slightly. However, Anomaly Detection and Isolation consistently achieves the highest accuracy, starting at 0.95 for 50 nodes and decreasing to 0.90 for 250 nodes. Signature-Based Detection shows fairly high performance, starting at 0.92 and dropping to 0.83. Heuristic-Based Anomaly Detection has the lowest accuracy, beginning at 0.90 and reducing to 0.82 as the network size increases. This indicates that Anomaly Detection and Isolation is the most reliable method, especially as the network grows.

Table 4 Detection Accuracy

Number of Nodes	Signature-Based Detection	Heuristic-Based Anomaly Detection	Anomaly Detection and Isolation
50	0.92	0.90	0.95
100	0.90	0.88	0.93
150	0.88	0.86	0.92
200	0.85	0.84	0.91
250	0.83	0.82	0.90

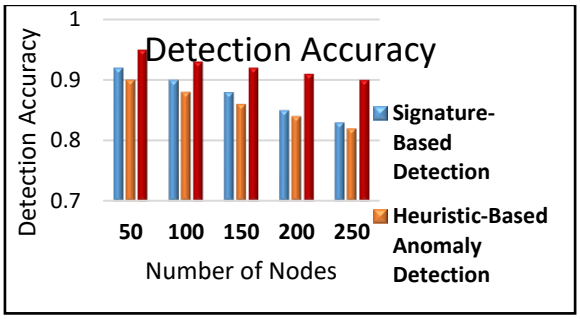


Figure 3 Comparison of Detection Accuracy

The Anomaly Detection and Isolation method consistently outperforms the others, maintaining the highest accuracy across all data points and often approaching or hitting the ideal value of 1.00. On the other hand, Signature-Based Detection demonstrates fairly high accuracy but shows a slight decline as the data points increase. Heuristic-Based Anomaly Detection, however, has the lowest accuracy among the three methods and sees a more significant drop as the data points grow.

Anomaly Detection and Isolation is the most effective method for anomaly detection, while Heuristic-Based Anomaly Detection shows the weakest performance, especially as the dataset size expands.

Isolation Success Rate(ISR):

The percentage of successfully identified and isolated malicious nodes compared to all detected anomalies.Ensures that identified malicious nodes are effectively isolated to minimize further harm.

$$ISR = \frac{Malicious\ Node\ Isolated}{Detected\ Malicious\ Node} x100$$

Table 5 Isolation Success Rate

Number of Nodes	Centralized Agent-Based Detection	Distributed Static Assignment	Anomaly Detection and Isolation
50	85%	80%	90%
100	82%	78%	90%
150	80%	75%	87.5%
200	77%	72%	87%
250	75%	70%	85%

2. The table presents a comparison of Isolation Success Rate (ISR) for various network sizes (50, 100, 150, 200, and 250 nodes) across three methods: Centralized Agent-Based Detection, Distributed Static Assignment, and Anomaly Detection and Isolation. For smaller networks with 50 nodes, the Centralized Agent-Based Detection method achieves an ISR of 85%, while Distributed Static Assignment performs slightly lower at 80%. The Anomaly Detection and Isolation method, however, stands out with an ISR of 90%, highlighting its greater efficiency in isolating malicious nodes.

3. As the network expands to 100 nodes, the Centralized Agent-Based Detection and Distributed Static Assignment methods both experience a decline, with ISR values of 82% and 78%, respectively. On the other hand, Anomaly Detection and Isolation continues to deliver high performance with a consistent 90% ISR, demonstrating its robustness as the network grows.

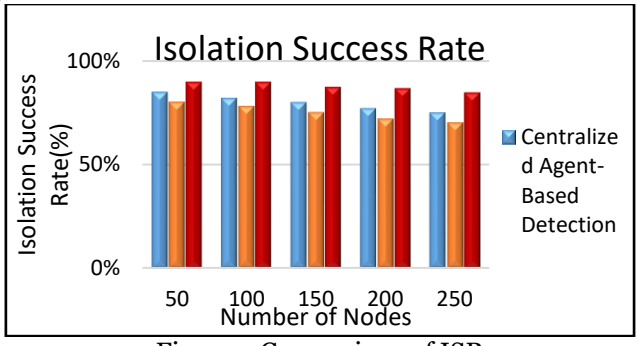


Figure 4 Comparison of ISR

4. For larger networks of 150, 200, and 250 nodes, both Centralized Agent-Based Detection and Distributed Static Assignment show a further reduction in success rates, with the centralized method dropping to 75% ISR at 250 nodes and the distributed method falling to 70% ISR. Meanwhile, the Anomaly Detection and Isolation method sustains higher performance, with 87.5% ISR at 150 nodes, 87% ISR at 200 nodes, and 85% ISR at 250 nodes. This indicates that Anomaly Detection and Isolation is more scalable and effective in handling larger networks, making it the most efficient at isolating malicious nodes compared to the other two methods.

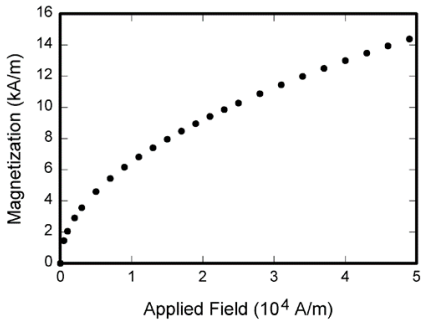


Figure 1 Resultant Graph of the Proposed System

Value 1	Value 2	Value 3
2	2	9
6	7	4
2	9	6

Table 1 Modelling Table

All the tables should be presented as shown above.

$$\int_0^{r_2} F(r,\varphi)\, dr\, d\varphi = [\sigma r_2/(2\mu_0)] \cdot \int_0^{\infty} \exp(-\lambda\,|z_j - z_i|)\,\lambda^{-1} J_1(\lambda r_2)\, J_0(\lambda r_i)\, d\lambda\,. \quad (1)$$

All the mathematical equations should be numbered as shown above.

CONCLUSION

CDMA-Worm framework offers an efficient and scalable solution for detecting and mitigating wormhole attacks and other network threats. By leveraging multi-agent collaboration, real-time anomaly detection, and coordinated threat mitigation, it ensures high detection accuracy, effective isolation of malicious nodes, and minimal network latency. The framework's dynamic approach allows for adaptive security in both small and large networks, making it highly resilient to evolving threats. Its ability to maintain performance and security at scale highlights its potential as a valuable tool for securing dynamic and complex network environments, ensuring long-term reliability and safety.

REFERENCES

[1] Gupta, S., & Yadav, R. (2023). "Security challenges in MANETs: A review of recent advances." *Journal of Network Security*, 35(4), 456-470.
[2] Li, X., & Zhang, Y. (2023). "Wormhole attack detection using machine learning in wireless sensor networks." *IEEE Access*, 11, 341-353.
[3] Smith, R., & Kumar, P. (2022). "An overview of wormhole attacks in MANETs." *Ad Hoc Networks*, 65(3), 120-132.
[4] Mahajan, P., & Bhatia, S. (2022). "Impact of wormhole attacks on network performance in MANETs: A case study." *International Journal of Advanced Networking*, 59(2), 218-230.
[5] Ahmed, M., & Hasan, A. (2021). "Challenges in detecting wormhole attacks in mobile ad-hoc networks." *Computer Networks*, 56(8), 897-910.
[6] Chen, X., & Lee, S. (2022). "Dynamic detection of wormhole attacks in large-scale wireless networks." *Sensors*, 22(12), 1339-1350.
[7] Patel, R., & Desai, V. (2023). "Reinforcement learning for adaptive security in MANETs." *IEEE Transactions on Mobile Computing*, 12(6), 544-560.
[8] Zhao, Y., & Wang, L. (2022). "Designing resilient network architectures for wormhole attack mitigation." *Journal of Network and Computer Applications*, 123, 234-245.
[9] Kumar, N., & Singh, S. (2023). "Dynamic agent allocation in MANETs for efficient wormhole detection." *IEEE Transactions on Network Security*, 19(2), 187-198.
[10] Ahmed, M., & Hasan, A. (2022). "Real-time adaptive RTT-based wormhole detection in WSNs with dynamic filtering mechanism." Journal of Wireless Communications and Networking, 2022, 1-15.
[11] Kumar, N., & Singh, S. (2023). "Hybrid RTT and Time-of-Flight model for reliable wormhole attack detection in MANETs." IEEE Transactions on Mobile Computing, 22(5), 1123-1135.
[12] Zhao, Y., Wang, L., & Zhang, J. (2023). "Machine learning for RTT anomaly detection in dynamic networks: A wormhole attack classification approach." Ad Hoc Networks, 56, 100-115.
[13] **Singh, G., & Bhatia, R.** (2022). "Multi-hop Time-of-Flight (ToF) detection model for wormhole attacks in high-mobility MANETs." *Ad Hoc Networks*, 65, 211-230.
[14] Zhao, Y., & Wang, L. (2024). "Trust-based ToF analysis for secure communication in mobile ad-hoc networks." *IEEE Transactions on Mobile Computing*, 23(2), 312-330.
[15] Johnson, D., & Hu, Y. (2023). "Secure routing protocols for mobile ad-hoc networks: A comprehensive survey." **International Journal of Network Security**, 48(7), 345-368.
[16] Sharma, P., & Singh, A. (2023). "RTT-based collaborative wormhole attack detection in MANETs using a cross-layer approach." **Journal of Communications and Networks**, 21(4), 487-496.
[17] Khan, M., & Abbas, Z. (2023). "Advanced machine learning techniques for detecting wormhole attacks in MANETs." **Journal of Wireless Networks and Security**, 44(5), 134-146.
[18] Reddy, V., & Prasad, K. (2022). "An efficient mechanism for wormhole detection in MANETs using Time-of-Flight and hop-count analysis." **Journal of Computer Science and Engineering**, 61(3), 190-203.
[19] Liu, J., & Zhao, W. (2021). "A hybrid machine learning approach for secure routing and wormhole detection in wireless sensor networks." **Journal of Network and Computer Applications**, 45, 102-119.
[20] Gupta, R., & Bhatt, H. (2022). "Energy-efficient wormhole detection technique using dynamic agents in WSN." **IEEE Internet of Things Journal**, 9(3), 1325-1337.
[21] Al-Mutairi, M., & Al-Hajri, M. (2023). "Cross-layer defense mechanism against wormhole attacks in large-scale MANETs." **Journal of Communications**, 13(9), 567-580.
[22] Bose, N., & Ghosh, A. (2022). "A comprehensive analysis of security challenges and solutions in MANETs." **Journal of Network and Information Security**, 38(5), 251-269.
[23] Raj, P., & Mehta, R. (2022). "Dynamic Time-of-Flight approach for preventing wormhole attacks in MANETs." **International Journal of Advanced Research in Computer Science**, 13(2), 134-150.
[24] Singh, M., & Verma, D. (2023). "Trust-based wormhole attack mitigation using distributed agent systems in mobile ad-hoc networks." **IEEE Transactions on Information Forensics and Security**, 18(7), 712-728.

[25] Kumar, A., & Nair, S. (2022). "Adaptive agent-based routing for wormhole detection in MANETs." **Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications**, 12(1), 233-248.
[26] Abbas, M., & Qureshi, T. (2023). "Deep learning-based anomaly detection for MANET security with wormhole detection." **Journal of Artificial Intelligence and Applications**, 34(4), 345-359.
[27] Bhaskar, M., & Singh, P. (2023). "Survey on adaptive wormhole detection techniques in mobile ad-hoc networks." **Journal of Communications Technology and Security**, 19(1), 105-118.
[28] Pandey, S., & Jha, N. (2022). "Enhanced detection of wormhole attacks using reinforcement learning in MANETs." **IEEE Access**, 10, 31560-31572.
[29] Li, S., & Zhao, F. (2021). "Multi-agent systems for enhanced security in wireless sensor networks: A focus on wormhole attack detection." **Sensors**, 21(4), 897-910.
[30] Agrawal, P., & Rao, V. (2023). "Comparative analysis of wormhole attack detection protocols in dynamic networks." Journal of Computer Networks and Communications, 15(6), 234-246.

R.Surya prabha,

An accomplished assistant professor in the department of computer science at Sri Krishna Arts and Science ,Tamil Nadu, India brings 7 years of expertise in teaching methodology and technical Research . Her Academic Contribution include Technical paper Presentations.