Volume 11, Issue 8, August – 2026 International Journal of Innovative Science and Research Technology
ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

Reducing Interception Latency in High-Security Facilities via AO* Optimization Algorithm

Sourya Dipta Ghosal[1]; Buddhadeb Sau[2]

A Thesis Submitted in Partial Fulfillment of the Requirements for the Award of the Degree of Master of Science (M.Sc.)

Mathematics

[2]Under the Supervision

[1]Examination Roll No.: PMAH0264049
[1]University Roll No.: 002420602007
[1]Registration No.: 159768 of 2021-2022

[2]Professor

[1]Department of Mathematics Jadavpur University
[2]Department of Mathematics Jadavpur University Academic Session

Publication Date: 2026/08/17

How to Cite: Sourya Dipta Ghosal; Buddhadeb Sau (2026) Reducing Interception Latency in High-Security Facilities via AO* Optimization Algorithm. *International Journal of Innovative Science and Research Technology*, 11(8), 701-749. https://doi.org/10.38124/ijisrt/26aug221

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

TABLE OF CONTENTS

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

DECLARATION

Is an original piece of research work carried out by me under the supervision of Prof. Buddhadeb Sau during the academic session 2025–2026. I further declare that,

This dissertation has not previously been submitted, either in full or in part, for the award of any degree, diploma, or any other academic qualification at this or any other University or Institution. The work presented in this dissertation is entirely my own except where due acknowledgement has been made in the text. All sources of information, published literature, books, journals, conference proceedings, online resources, and other references have been properly cited and acknowledged. The research findings presented in this dissertation are genuine and have been obtained through independent investigation and analysis. I understand that any instance of plagiarism, data fabrication, or academic misconduct discovered at any stage shall render this dissertation liable for cancellation and may invite disciplinary action in accordance with the rules and regulations of the University.

I hereby affirm that the above statements are true to the best of my knowledge and belief.

Place: Kolkata, India Sourya Dipta Ghosal

Date:

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

CERTIFICATE FROM THE GUIDE

During the academic session 2025–2026, is a bonafide record of the original research work carried out by her under my supervision and guidance.

I further certify that this dissertation has not been submitted, either in whole or in part, to any other University or Institute for the award of any degree or diploma.

Place: Kolkata

Date:

Prof. Buddhadeb Sau
Supervisor
Department of Mathematics Jadavpur University

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

CERTIFICATE

This is to certify that the dissertation entitled “Reducing Interception Latency in High-Security Facilities via AO* Optimization Algorithm” is a bonafide record of the study/work carried out by Sourya Dipta Ghosal, Roll No. PMAH0264049, during the academic year 2025–2026, under the guidance of Prof. Buddhadeb Sau, Department of Mathematics, Jadavpur University.

Signature of the Guide

(Prof. Buddhadeb Sau)

Place: Kolkata, India

Date:

Head of the Department

(Name and Official Seal)

Place: Kolkata, India

Date:

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

CERTIFICATE OF APPROVAL

This is to certify that the dissertation entitled "Reducing Interception Latency in High-Security Facilities via AO* Optimization Algorithm" submitted by Sourya Dipta Ghosal has been approved in quality and volume. It has been found fit for the partial fulfillment of the requirements for the award of the degree of Master of Science (M.Sc.) in Mathematics.

Internal Guide

Signature: _____

Name and Official Seal: _____

Place: Kolkata, India Date: _____

External Examiner 1

Signature: _____

Date: _____

Name: _____

Designation: _____

Department: _____

Institute and Address: _____

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

ACKNOWLEDGEMENT

It is with immense pleasure and deep gratitude that I express my sincere thanks to all those who have contributed directly or indirectly towards the successful completion of this dissertation entitled, “Reducing Interception Latency in HighSecurity Facilities via AO* Optimization Algorithm”. First and foremost, I express my heartfelt gratitude to my respected supervisor, Prof. Buddhadeb Sau, Department of Mathematics, Jadavpur University, for providing invaluable guidance, constant encouragement, constructive criticism, and continuous motivation throughout the course of this research. Their profound knowledge, patience, and unwavering support have greatly enriched my understanding and inspired me to complete this work successfully.

I express my sincere gratitude to the Head of the Department, Dr. Subhas Chandra Mandal, Department of Mathematics, for providing the necessary facilities and creating an excellent academic environment for carrying out this research work.I would also like to express my heartfelt thanks to all the faculty members of the Department of Computer Science and Engineering for sharing their valuable knowledge and providing academic support whenever required.

My sincere thanks are also due to my classmates and friends for their continuous encouragement, insightful discussions, and moral support throughout the completion of this work. I owe my deepest gratitude to my beloved parents and family members whose unconditional love, blessings, sacrifices, patience, and encouragement have always been the greatest source of inspiration in every stage of my life.

Without their constant support, this work would not have been possible.

Finally, I thank the Almighty for blessing me with good health, strength, perseverance, and determination to successfully complete this dissertation.

Sourya Dipta Ghosal
Roll No.: PMAH0264049
Department of Mathematics
(Jadavpur University)

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

ABSTRACT

Traditional physical security systems are largely reactive, relying on localized alarms that trigger only after a perimeter breach has occurred. In complex, high-value environment – such as data centers or nuclear facilities – this latency often allows intruders to reach their objectives before an interceptor can arrive. This thesis proposes a proactive security framework that models facility logic as an AND-OR graph, representing potential "attack recipes" as a series of interdependent security states. By employing the AO* optimization algorithm, the system dynamically identifies the most probable paths an intruder might take based on real-time Anomaly detection from distributed sensor networks. To ensure the integrity of the decision-making process against digital tampering, the framework integrates Elliptic Curve Digital Signature Algorithms (ECDSA) to authenticate every state change within the graph. The core of this research investigates whether AO* optimization can significantly reduce the response time of autonomous mobile interceptors (drones or robots) by positioning them at "logic bottlenecks" – points in the AND-OR graph that an intruder must satisfy to complete their objective. Preliminary simulation suggests that this predictive allocation outperforms traditional patrolling methods, providing a robust, self-healing, and mathematically verifiable approach to modern autonomous defense.

Keywords: *AO* Algorithm, Interception Latency, High-Security Facilities, Artificial Intelligence, Optimization, Cybersecurity, Elliptic Curve Digital Signature Algorithms, Risk Based Pathfinding, Anomaly Detection*.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

CHAPTER ONE
INTRODUCTION

➢ *Overview*

Traditional physical security systems are largely reactive, relying on localized alarms that trigger only after a perimeter breach has occurred. In complex, high-value environment – such as data centers or nuclear facilities – this latency often allows intruders to reach their objectives before an interceptor can arrive. This thesis proposes a proactive security framework that models facility logic as an AND-OR graph, representing potential "attack recipes" as a series of interdependent security states. By employing the AO* optimization algorithm , the system dynamically identifies the most probable paths an intruder might take based on real-time Anomaly detection from distributed sensor networks. To ensure the integrity of the decision-making process against digital tampering, the framework integrates Elliptic Curve Digital Signature Algorithms (ECDSA) to authenticate every state change within the graph. The core of this research investigates whether AO* optimization can significantly reduce the response time of autonomous mobile interceptors (drones or robots) by positioning them at "logic bottlenecks" – points in the AND-OR graph that an intruder must satisfy to complete their objective. Preliminary simulation suggests that this predictive allocation outperforms traditional patrolling methods, providing a robust, self-healing, and mathematically verifiable approach to modern autonomous defense.

The landscape of physical security is undergoing paradigm shift. As high-value facilities – ranging from decentralized data centers to autonomous industrial hubs – become more complex, the methods used to protect them must evolve from reactive measures to proactive, intelligent systems. Traditional security relies on isolated sensors and human intervention, a combination that often suffers from high latency [26] and "siloed" data. When an alarm is triggered, the response is typically retrospective: security personnel or autonomous units move to the site of the breach after the event has occurred. This thesis introduces Sentinel Logic, a framework that treats security not as a set of perimeters, but as a series of logical states. By modelling a facility's security architecture as an AND-OR graph, we can represent an intruder's progress as a sequence of satisfied conditions.

➢ *Problem Statement*

In multi-level, high-security environments, the "path" an intruder takes is rarely a straight line. It is a contingent strategy. For instance, gaining access to a server room might require a physical breach (Door A) AND a digital bypass (Terminal B). Standard path-finding algorithms like A* are designed to find the shortest spatial distance, but they fail to account for these logical dependencies. Furthermore, autonomous interceptors (drones or ground robots) often operate on static patrol routes. This creates "blind spots" that savvy intruders can exploit. There is a critical need for a system that can:

- Predict the most likely future states of a breach.
- Verify that sensor data hasn't been tampered with (preventing "ghost" or "hidden" paths).
- Optimize the deployment of interceptors to meet the intruder at logical bottlenecks rather than chasing their shadows.

➢ *Indian Interception Latency Scenario for Securing High-Value Assets*

India's High-Value Assets (HVAs), including defence installations, nuclear facilities, strategic communication networks, airports, and critical information infrastructure, require rapid detection and response mechanisms to counter physical and cyberelectromagnetic threats. Interception latency—the elapsed time between threat detection and successful mitigation—is a critical performance metric that determines the effectiveness of security operations.

A typical AI-enabled interception framework operates through three latency phases:

- Threat Detection (0–15 ms): Distributed sensors, radar, and AI-based intrusion detection systems identify anomalies such as electronic reconnaissance, RF jamming, UAV incursions, or unauthorized access.
- Dynamic Defence and Secure Communication (15–50 ms): Adaptive frequency hopping, secure routing, post-quantum cryptography, and encrypted communication channels preserve the confidentiality and integrity of mission-critical data while AI evaluates the optimal response strategy.
- Active Response (50–3000 ms): Electronic countermeasures, counterUAV systems, or rapid response teams are deployed to neutralize the identified threat.

By modelling alternative response strategies using the AO* optimization algorithm, the system selects the minimum-cost interception path based on threat severity, response time, communication delay, and resource availability. This AIdriven framework significantly reduces interception latency, enhances operational resilience, and strengthens the protection of India's critical infrastructure against both physical and cyber-electromagnetic threats.

Note: The latency ranges (0–15 ms, 15–50 ms, and 50–3000 ms) are illustrative conceptual values suitable for a research framework or simulation model rather than verified operational timings for Indian security systems.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

- *Proposed Solution*

This research proposes an integrated autonomous defense system. At its core, the AO* optimization algorithm is utilized to navigate a dynamic AND-OR graph of the facility. Unlike traditional search methods, AO* can handle the "AND" conditions inherent in high-security logic. To bridge the gap between the physical and digital worlds, Machine-Learning based Anomaly Detection serves as the graph's sensory input, while Digital Signature Algorithms (DSA) provide a cryptographic "Trust layer". This ensures that every state transition – every "opened door" or "accessed rack" – is mathematically verified, preventing attackers from "blinding" the system through log manipulation.

- *Significance of the Study*

By shifting the focus from "where the intruder is" to "what the intruder must do next," this framework enables a Predictive Interception model. This is particularly vital for the security of Agentic AI and decentralized systems where human oversight is minimal. The results of this study will provide a blueprint for "zero-trust" physical security architectures where the building itself becomes an active, thinking participant in its own defence.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

CHAPTER TWO
LITERATURE REVIEW

- *Evolution of Pathfinding: From A* to AO**

Traditional path finding in robotics and security has long relied on algorithms like Dijkstra and A*. These are designed to ind the shortest distance between two points in a state-space. However, these models operate on OR-only graphs, where each edge represents a single choice (e.g., ”Go Left” OR ”Go Right”)[3, 5].

In high-security threat modelling, a breach is rarely a single path; it is a contingent problem. The AO* algorithm (And-Or Star) represents a significant advancement because it handles ”Problem Reduction.” It decomposes a top-level goal (e.g., ”Exfiltrate Data”) into sub-goals that must all be met (AND) or alternative methods (OR) [9].

This study builds upon the work of , applying their heuristic search principles to the physical security of autonomous interceptors [10].

Moreover, AO* algorithm specially designed to adapt to changes without initiating a new search. It excels in situations with uncertainty, quickly adjusting plans in response to new information rather, A* fails in this situation of uncertainty. AO* is highly suitable for real-time applications with dynamic elements [24]. A* algorithm requires a consistent heuristic for optimal results, whereas, AO* doesn’t strictly require a consistent heuristic. That means, AO* offers more flexibility in heuristic choice as it builds an optimal solution graph for dependent subproblems using a dynamic, top-down/bottom-up cost revision approach that works effectively even with non-admissible or less consistent heuristics [11].

- *Autonomous Interception and Mobile Robotics*

Current research in autonomous security robotics primarily focuses on area coverage and patrol routing. Dickens et al.,[17] proposed a robot platform HERMES.

Which autonomously patrol outdoor areas performing surveillance and providing automated alerts of detected vehicles and people. It is designed to autonomously patrol outdoor areas performing surveillance and providing automated alerts of detected vehicles and people. The design and testing of this system are covered in this paper. The design philosophy focused on the use of off-the-shelf components wherever possible with the base of the robot being a modified electric quadbike. The testing has verified that the surveillance robot can perform real-time person and vehicle detection, video streaming, manual and autonomous navigation on a low-cost platform.

Methods such as “Ant Colony Optimization” aim to minimize the time any single point remains unobserved. H. Calvo et al., proposed an improved version of Ant Colony Optimization algorithm (ACO) i.e., Max Min Ant System (MMAS) which allows to allocate an optimal number of human and material for patrolling and optimization of patrolling routes for personnel working in public security. However, these methods are “intruder-agnostic” – they do not account for the logical progress an intruder has already made. This thesis seeks to bridge the gap by using the output of the AO* search to “pre-calculate” interception points, shifting the robotics focus from patrolling to anticipating.

- *Machine Learning in Anomaly Detection*

The transition of nodes in a security graph requires high-fidelity triggers. Recent literature highlight the success of Recurrent Neural Networks (RNN) and Autoencoders in identifying deviations from “baseline” environmental noise. S.Narmadha et al. [28], Their DL model combines the LSTM with the Autoencoder model, where the auto-encoder learns normal patterns. The performance of their DL model is evaluated with existing IDS methods and the results show that the proposed model achieves maximum detection accuracy rate of 0.9989. In a high-security context, an anomaly is not just a loud noise; it is a pattern (e.g., a specific vibration frequency on a server rack). This research integrates these ML triggers as the “edge providers” that feed real time data into the AO* graph.

Through case studies in IT and financial technology, we illustrate the effectiveness of ML-driven anomaly detection in network security and fraud prevention. The transformative potential of ML for anomaly detection is confirmed, advocating a proactive approach in its implementation while addressing ethical and practical challenges.Video Surveillance systems, especially with edge computing technology, benefit from ML-based anomaly detection for realtime security monitoring. The adaptability of ML technologies is crucial for addressing new challenges in this dynamically evolving field [35].

- *Cryptographic Integrity in IoT Systems*

As security systems move toward decentralized, autonomous architectures, they become vulnerable to “data injection” attacks. If an intruder can spoof a “System Normal” signal while moving through a facility, the AI remains blind. The Digital Signature Algorithm (DSA) [koblitz1998] [1] is a well-established standard for ensuring data authenticity [36].

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

While traditionally used in web traffic, this thesis explores its application in real-time state verification, ensuring that each node "solved" in the AO* graph is backed by a verifiable, hardware-level signature. With the proliferation of Internet of Things (IoT) applications, ensuring data integrity has become a paramount concern. The decentralized nature of these systems, coupled with the increasing volume and velocity of data, poses significant challenges for maintaining trust and reliability [36]. It delves into innovations such as homomorphic encryption, blockchain integration, lightweight cryptography, and zero-knowledge proofs, highlighting their potential to enhance security without compromising system efficiency.

To secure the IoT environment, cryptographic approaches are crucial. They [37] are employed to guarantee the CIA the information exchanged via networks. In Internet security, cryptographic algorithms are used to encrypt data and prevent unauthorised access, ensure data integrity by detecting any modifications during transmission, and provide authentication so that only individuals with permission can get information. By using techniques for encryption of the data, cryptography is the art and science of keeping the information safe while it is being carried across a network.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

CHAPTER THREE
METHODOLOGY

➢ *Graph Construction*

The facility is modelled as a Directed Acyclic Graph (DAG) where:

- *Leaf Nodes*: Represent raw sensor inputs
- *AND Nodes*: Represent security checkpoints requiring multiple failures (e.g., "Biometric Scan Passed" AND "Physical Door Opened")
- *OR Nodes*: Represent alternative routes for the intruder
- *Goal Node*: The "High-Value Asset" (e.g., The Central Server)

➢ *AO* Heuristic Function*

The efficiency of the search depends on the heuristic $h(n)$. In this framework, $h(n)$ is calculated based on [5, 11] :

- Physical Distance: The meters remaining to the target
- Security Difficulty: The estimated time required to bypass the next logical "AND" gate.
- Interceptor Proximity: A dynamic weight that increases the cost of a path if a drone is already nearby, forcing the algorithm to evaluate if the intruder will pivot to an alternative OR-path.

$$h(n) = \text{Physical Distance} + \text{Security Difficulty} + \text{Interceptor Proximity} \quad (1)$$

➢ *Cryptographic Handshake*

Each sensor is equipped with a unique private key. When a threshold is met:

- The sensor generates a status packet: Status: Triggered | Timestamp: T1 .
- The packet is hashed and signed:

$$S = \text{Sign}_{priv}(\text{Hash}(\text{Packet})) \quad (2)$$

- The central controller verifies the signature using the sensor's public key before updating the AO* graph state [29].

➢ *Main Aim of the Proposed Methodology*

The primary aim of the proposed methodology is to minimize the interception latency in high-security facilities by intelligently predicting an intruder's most probable path and deploying interception resources in the shortest possible time. Traditional surveillance systems generally rely on static rule-based responses, where security personnel or autonomous agents react only after an intrusion event has been fully confirmed. Such approaches often introduce significant delays in threat detection, path analysis, and response coordination.

To address these limitations, the proposed framework models the entire security environment as an AND-OR graph represented by a Directed Acyclic Graph (DAG). This representation captures the logical dependencies between multiple security checkpoints, alternative intrusion routes, and critical assets within the facility. The AO* optimization algorithm is then employed to dynamically identify the minimumcost path that an intruder is most likely to follow.

The methodology aims to achieve the following objectives:

- Early Intrusion Prediction: Detect and predict potential intrusion paths before the attacker reaches high-value assets by continuously analyzing sensor activations and graph state transitions.
- Reduction of Interception Latency: Minimize the response time between intrusion detection and interceptor deployment through heuristic-guided search and real-time decision making.
- Optimal Resource Allocation: Determine the most effective positioning and movement of autonomous drones or security agents by selecting the leastcost interception strategy.
- Adaptive Threat Assessment: Dynamically update the estimated intrusion path based on changing environmental conditions, newly activated sensors, and interceptor positions.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

- Secure Sensor Communication: Ensure the authenticity and integrity of sensor-generated alerts through cryptographic signatures and public-key verification mechanisms.

- Elimination of False Alarms: Prevent malicious or compromised sensors from injecting fraudulent intrusion events into the surveillance system.

- Scalability for Large Facilities: Enable efficient operation in complex infrastructures such as airports, military bases, research laboratories, data centers, and smart factories where multiple intrusion paths exist simultaneously.

Therefore, the proposed methodology combines graph-based reasoning, heuristic optimization, and cryptographic authentication to create an intelligent and secure surveillance framework capable of reducing interception latency while maximizing protection of critical assets.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

CHAPTER FOUR
SYSTEM ARCHITECTURE

➢ *The Tri-Layered Framework*

The proposed system, sentinel logic, operates through three integrated layers that bridge the gap between physical sensors and autonomous decision-making [30,31].

- *Perception Layer (Sensors and ML)*

The "eyes and ears" of the facility consist of a distributed network of IoT sensors – acoustic, thermal, vibration and motion [31].

✓ *Edge Processing*: Each sensor node runs a lightweight Anomaly Detection model (such as an Isolation Forest or a pruned Autoencoder).
✓ *Trigger Mechanism*: Instead of streaming raw data, which consumes bandwidth, sensors only broadcast when a "State change" is detected (e.g., vibration pattern matching a forced entry).

- *Logic Layer (The AO∗ Controller)*

This is the central "brain" where the AND-OR graph resides [31, 32].

✓ *Graph Synthesis*: The controller maintains a digital twin of the facility's security logic.
✓ *Path Optimization*: When a sensor triggers, the AO* algorithm re-evaluates the "Most Probable Threat Path" (MPTP). It accounts for the fact that an intruder might need to satisfy multiple conditions (AND) or choose between different routes (OR).
✓ *Heuristic Calculation*: The system calculates the "cost-to-goal" by factoring in physical distance, the number of security layers, and the current positions of all autonomous interceptors [32].

- *Action Layer (Autonomous Interceptors)*

The final layer consists of a leet of mobile units (Drones or Ground Robots):

✓ *Predictive Deployment*: Rather than responding to the last known location of an intruder, the action layer receives co-ordinates for the next logical bottleneck identified by the AO* controller.
✓ *Dynamic Re − tasking*: If the AO* algorithm detects "pivot" (the intruder moving toward a different OR branch), the interceptors are re-routed in realtime to intercept the new path [31, 32].

➢ *Cryptographic State Verification*

To prevent "Adversarial Logic Attacks" – where an intruder hacks the network to tell the that a door is "Closed" when it is actually "Open" – every communication between the perception and logic layers is secured using Digital Signature Algorithms (DSA).

- *Event Generation*: A sensor detects a breach at node N.
- *Signing*: The sensor signs the status S and a timestamp T using its unique private key:

$$\sigma = \text{Sign}_{Kpriv}(S||T) \quad (3)$$

- *Transmission*: The packet *S, T,* σ is sent to the controller.
- *V erification*: The controller verifies σ using the sensor's public key. If the signature is invalid or the timestamp indicates a "replay attack", the update is rejected, and a System Integrity Alert is triggered [29, 31].

➢ *Data Flow and Communication*

The system utilizes a Publish-Subscribe (Pub/Sub) architecture over a secure industrial protocol (like DDS or encrypted MQTT) (Ferraz Junior et al.) [18]. This ensures that the AO* controller can handle simultaneous inputs from hundreds of sensors without a linear increase in latency.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

Algorithm 1: Identification of Data Flows

```
Input: currentNode – NODE, prefix – seq(DataFlow)
Data : allSeqs – seq(seq(DataFlow)),
       visited – P(NODE),
       nodeFlows – NODE → DataFlow
1  Function dataFlows(currentNode, prefix):
2      visited ← visited ∪ {currentNode};
3      dfs ← nodeFlows(currentNode);
4      if dfs = ∅ then
5          if |prefix| > 0 then
6              allSeqs ← allSeqs ∪ {prefix};
7      else
8          foreach df ∈ dfs do
9              newPrefix ← prefix;
10             append df to newPrefix;
11             if df.to ∈ visited then
12                 allSeqs ← allSeqs ∪ {newPrefix};
13             else
14                 dataFlows(df.to, newPrefix);
15     return;
```

Table 1 Communication Protocols and Latency Requirements

Segment	Protocol	Data Types	Latency Requirement
Sensor → Edge	12C/SPI	Raw Signal (bits)	≤ 1 ms
Edge → Controller	MQTT/DDS (TLS 1.3)	JSON (Signed State)	≤ 10 ms
Controller → Drone	MAVLink/ ROS2	Geometry Messages	≤ 5 ms
Drone → Controller	Telemetry	State/Position Vectors	Continuous

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

CHAPTER FIVE
GRAPH-BASED ANOMALY DETECTION

➢ *Introduction to GBAD*

Graph-Based Anomaly Detection (GBAD) represents a sophisticated approach to identifying unusual patterns by analyzing the structural relationships within graph databases. This method, formalized by Akoglu et al. [6], excels at detecting rare or significantly different graph objects (nodes, edges, or substructures) by leveraging the inherent interconnections among data elements. As highlighted by Sensarma et al. [7], the increasing ubiquity of graph data has sparked numerous studies exploring its potential in anomaly detection, particularly due to its' ability to capture long – range correlations between objects. GBAD methodologies can be classified based on the type of anomalous component (i.e., node, edge, subgraph), graph type (i.e., static, dynamic) [7, 8], method used (i.e., probabilistic/statisticalbased, matrix/tensor decomposition-based, and distance/similarity-based methods), and anomaly type (i.e., sparse anomaly, group anomaly, sudden anomaly, gradual anomaly). The core formulation of the GBAD methods typically begins with a graph defined as.

$$G = (V, E, \gamma) \quad (4)$$

Where V is the set of nodes, E is the set of edges and γ is the set of labels representing the normal or anomalous states of graph elements (nodes, edges, or the entire graph). The goal of anomaly detection is to learn a mapping function for one of the following tasks: Node-level Anomaly Detection: $f : V \rightarrow \gamma$, Edge-level Anomaly Detection: $f : E \rightarrow \gamma$, Graph-level Classification: $f : G \rightarrow \gamma$ The implementation of GBAD follows a structured methodology, comprising graph construction, embedding and detection phases. Li et al. [19] exemplified this approach through a log anomaly detection system that incorporates five key steps: log parsing, grouping, graph construction, representation learning, and anomaly detection.

Anomaly Detection Approaches for Intrusion Detection: Intrusion Detection Systems (IDSs) and their classification into host-based and network-based approach, as well as signature-based and anomaly-based IDSs (A-IDSs) [10] are established in the literature.The field of cybersecurity is changing dramatically in this dynamic age of digital revolution. This work on Anomaly Detection in Cybersecurity using GraphBased Approaches represents a ground- breaking project that uses Graph Neural Networks' (GNNs'), Graph-Based Behavioural Anomaly Detection (GBBAD), Behavioural Identification Graph (BIG) and Graph-Based Botnet Detection (GBBD) capabilities to revolutionize the way we defend our digital borders [38].

➢ *System Overview*

The framework is designed to be durable, egalitarian, and intelligible in nature. Graph theory provides a solution through using graph-based techniques. Therein, the nodes represent entities while edges or links resemble the interactions among those said entities [38].

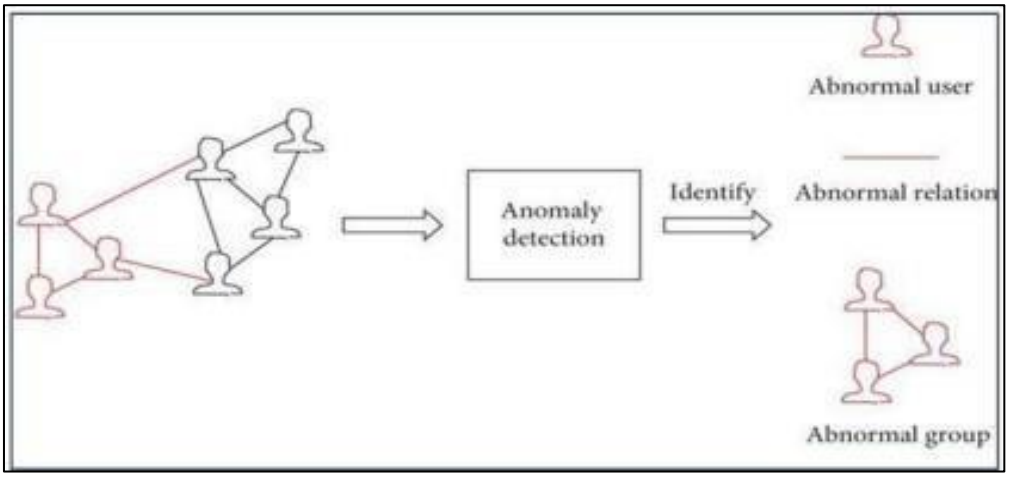

Fig 1 Schematic Diagram of Graph-Based Anomaly Detection

The quality and relevance of the dataset greatly affect anomaly detection efficiency. Importantly, our anomaly detection system is guaranteed of being wellrounded as well as contextually conscious by first converting unstructured data into structured graphs for its further assessment. Furthermore, smooth interaction with existing cybersecurity infrastructures leads to timely sharing of threat intelligence and defense mechanisms.

➢ *Comparison Between GBAD and FBAD*

In a real computing environment, Mirai, Black Energy, Zeus, Athena and Ares have recently proliferated into five botnets and a considerable amount of network traffic is gathered [39]. Many experiments have shown that BotMark is very effective. It achieves

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

99.49 % accuracy with flow-based detector, while a graph-based detector has 91.66 % detection rate. By using a hybrid of both detectors, BotMark surpasses the performance of each individual detector with a 99.94 % detection accuracy.

However the graph-based approach makes up for what the Cflow based detector cannot detect by finding certain Zeus bots which are not detected by it. Clearly through hybrid analysis of traffic behaviors based on graphs and flows, identifying botnets becomes both essential and vital [39].

➢ *GBAD Workflow for Intrusion Detection*

This section presents a systematic framework for graph-based anomaly detection methods, comprising six distinct phases identified through our literature review: data capturing, graph construction, graph pre-processing, graph anomaly detection, performance evaluation, and post-detection analysis. Figure 2 provides a visual representation of these sequential phases [40].

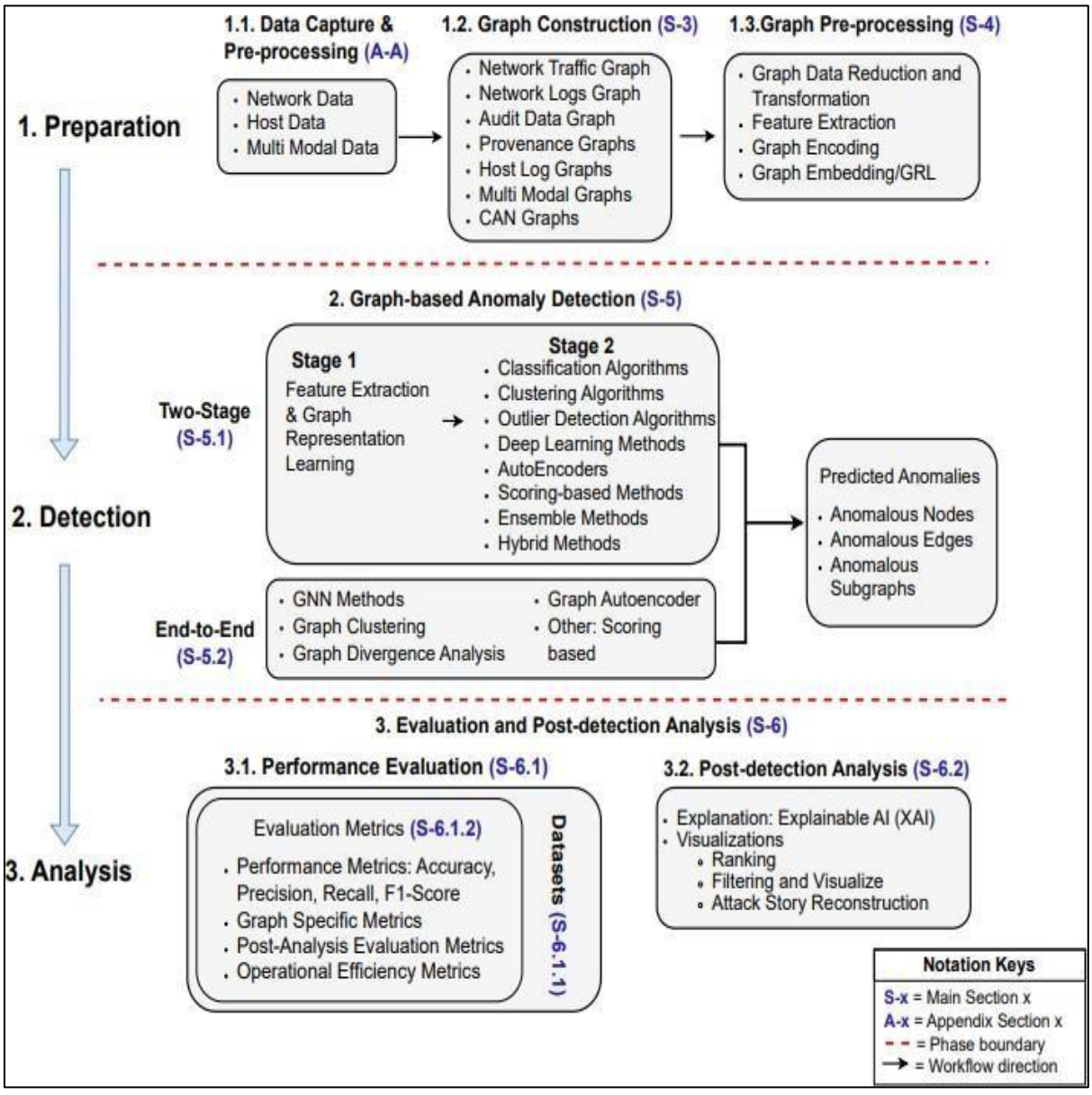

Fig 2 GBAD Workflow for Intrusion Detection

The intrusion detection process begins with data capturing, where infrastructure data from various sources undergoes initial pre-processing to ensure suitability for detection purposes. This processed data is then transformed into graph representations in the graph construction phase. The graph pre-processing phase encompasses several critical sub-steps: graph data reduction, graph data transformation, graph feature extraction and feature generation, graph encoding, and GRL. The anomaly detection phase employs either two-stage or end-to-end detection approaches. Following detection, the post-analysis phase focuses on explaining and visualizing the identified anomalies. The final phase involves evaluating the detection results using established metrics and benchmark datasets [39].

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

CHAPTER SIX
DIGITAL SIGNATURE ALGORITHM

➢ *History of DSA Algorithm*

In 1991 the U.S. government's National Institute of Standards and Technology (NIST) proposed a Digital Signature Standard (DSS) based on a certain Digital Signature Algorithm (DSA). The role of DSS is expected to be analogous to that of the older Data Encryption Standard (DES): it is supposed to provide a standard digital signature method for use by government and commercial organizations. But while DES is a classical ("private key") cryptosystem [2], in order to construct digital signatures it is necessary to use public key cryptography. NIST chose to base their signature scheme on the discrete log problem in a prime finite field F_p. The DSA is very similar to a signature scheme that was originally proposed in [Schnorr 1990] [21]. It is also similar to a signature scheme in [El Gamal 1985a] [13]. We now describe how the DSA works.

➢ *The DSA Algorithm Procedure*

The signature process in the DSA algorithm process is shown in figure 1, and the validation process is shown in figure 2 below [1].

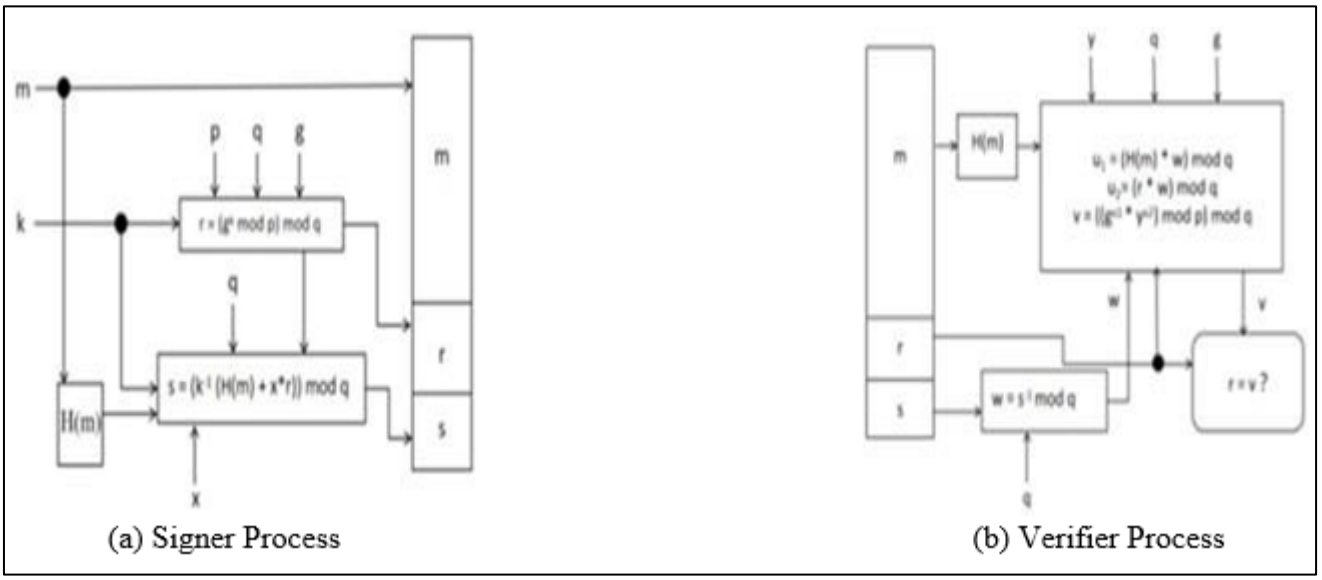

Fig 3 (a) Signer Process & (b) Verifier Process

➢ *Preready*

- Choose a prime q of about 160 bits (to do this, use a random number generator and a primality test) [1].
- Choose a second prime $p \equiv 1 \pmod{q}$ and has about 500 bits (more precisely, the recommended number of bits is a multiple of 64 between 512 and 1024).
- Choose a generator g of the unique cyclic subgroup F_p of order q (Do this by computing

$$g_0^{\frac{p-1}{q}} \pmod{p} \tag{5}$$

For a random integer g_0; if this number is not equal to 1, it will be a generator).

- She takes a random integer x in the range $0 < x < q$ as her secret key, and sets her public key equal to

$$y = g^x \pmod{p} \tag{6}$$

➢ *Key Generation*

Private Key is x where $0 < x < q$.

Public Key is y where

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

$$y = g^x \pmod p \tag{7}$$

Combined public key (p, q, g, y).

➢ *Signature*

- After obtaining the information m, first randomly obtain a number k, so that k satisfies: $0 < k < q$ [1].
- Calculate the r in the signature of the information m so that r satisfies:

$$r = (g^k \bmod p) \bmod q \tag{8}$$

- Calculate s in the signature of the information m, so that s satisfies:

$$s = k^{-1}(H(m) + xr) \bmod q \tag{9}$$

Where $H(x)$ function is the one-way Hash function and the $H(m)$ is the hashing of the message m.

- Get the signature of the information m: (r, s).

➢ *Verification*

- After receiving the information m, the received signature (r, s) is calculated.
- Range validation of r and s, if r satisfies: $0 < r < q$ and $0 < s < q$.
- First calculate the value of w, so that w satisfies:

$$w = s^{-1} \bmod q \tag{10}$$

- Calculate the index u_1 of g, send u_1 satisfied:

$$u_1 = wH(m) \bmod q \tag{11}$$

- Calculate the index of u_2 of the y, send u_2 satisfied:

$$u_2 = wr \bmod q \tag{12}$$

- Calculate the verification value v so that v satisfies:

$$v = g^{u1}y^{u2} \pmod p \pmod q \tag{13}$$

- If $v = r$, the validity of the signature can be verified.

This signature scheme has the advantage that signatures are fairly short, consisting of two numbers of 160 bits (the magnitude of q). On the other hand, the security of the system seems to depend upon intractability of the discrete log problem in the multiplicative group of the rather large field F_p. Although to break the system it would suffice to find discrete logs in the smaller subgroup generated by g, in practice this seems to be no easier than finding arbitrary discrete logarithms in F_p . Thus the DSA seems to have attained a fairly high level of security without sacrificing small signature storage and implementation time.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

CHAPTER SEVEN
ELLIPTIC CURVE CRYPTOGRAPHY AND ECDSA

➢ *Need for ECDSA*

In recent years, Elliptic Curve Cryptography (ECC) has attracted the attention of researchers and product developers because of its robust mathematical structure and highest security in comparison to other existing algorithms like RSA (Rivest Adleman and Shameer Public key Algorithm). Elliptic Curve Digital signature represents one of the most widely used security technologies for ensuring un-forgeability and non-repudiation of digital data. Its performance heavily depends on an operation called point multiplication. Furthermore, root cause of security breakdown of ECDSA is that it shares three points of the elliptic curve public ally which makes it feasible for an adversary to gauge the private key of the signer. In this paper we proposed a new ECDSA which involves not as much of point multiplication operations as in existing ECDSA and shares only two curve points with everyone. The proposed method also reduces the point addition and point doubling operations. It is found to be more secure in contrast to existing ECDSA [41].

➢ *ECC Encryption*

Suppose that Alice and Bob want to communicate with each other. They should follow the next steps [1, 12]:

- Alice selects a, b, p to create an elliptic curve $E_p(a, b)$ and choose a point M as base point.
- Alice selects a private key y and gets public key $Y = yM$. Then Alice sends $E_p(a, b)$, Y and M to Bob.
- Bob receives the message, encode message L to a point N in elliptic curve $E_p(a, b)$. Next, Bob chooses a private key $r < n$.
- Bob computes

$$C_1 = N + rY \quad (14)$$

$$C_2 = rM \quad (15)$$

- *Bob sends C_1,C_2 to Alice.*

After receiving Bob's message, Alice decrypts the message by computing $N = C_1 - yC_2$ to obtain the plaintext.

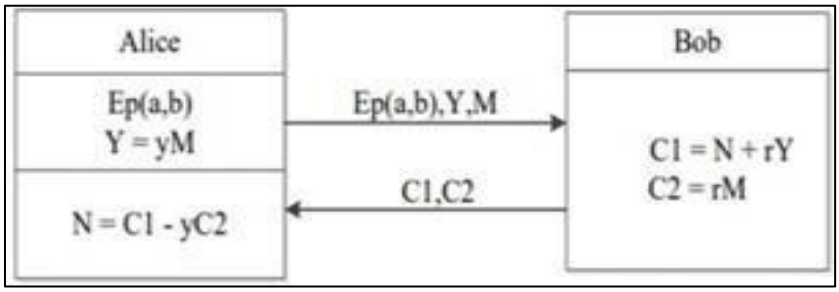

Fig 4 ECC Encryption and Decryption

➢ *ECDSA Algorithm*

In 1992, Vanstone proposed the Elliptic Curve Digital Signature Algorithm (ECDSA). The ECDSA is based on the elliptic curve password of the digital signature algorithm resimulation, its security is based on the problem of elliptic curve discrete log. However, the discrete logarithm problem of the elliptic curve is hardly a rising gradient compared to the discrete logarithm problem for DSA, so the unit bit intensity of the elliptic curve cryptography system is much higher than that of the traditional discrete logarithm system [1, 12, 13].

➢ *ECDSA Key Generation*

For simplicity, we shall use elliptic curves defined over a prime field F_p, although the construction can easily be adapted to other finite fields as well.

Let E be an elliptic curve defined over F_p, and let P be a point of prime order q in $E(F_p)$; these are system-wide parameters.

Note: As in DSA, q denotes not a power of p, but rather a different prime number. Unlike in the DSA, where q is much smaller than p, in the ECDSA, q is about the same size as p.

Each user selects a random integer x in the interval $1 < x < q - 1$ and computes $Q = xP$.

Therefore, public key is Q and private key is x.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

➢ *ECDSA Signature*

To sign a message *m*, user does the following:

- Select a random integer k in the interval $1 < k < q - 1$.
- Compute

$$kP = (x_1, y_1) \text{ and } r = x_1 \pmod{q} \tag{16}$$

(That is, x_1 is regarded as an integer between 0 *and* $p - 1$, and r is taken to be its least non-negative residue modulo q). If $r = 0$, return to step 1st step. If $r = 0$, then the signing equation.

$$s = k^{-1}(H(m) + xr) \bmod q \tag{17}$$

Does not involve the private key x; hence 0 is not suitable value r.

- Compute $k^{-1} \pmod{q}$
- Compute

$$s = k^{-1}(H(m) + xr) \bmod q \tag{18}$$

Where $H(m)$ is the hash value of the message. If $s = 0$, return to 1st step. If $s = 0$, then $s^{-1} \bmod q$, which is required in 3rd step. If k is chosen at random, then the probability that either $r = 0$ or $s = 0$ is negligibly small.

- The signature for the message m is the pair of integers (r, s).

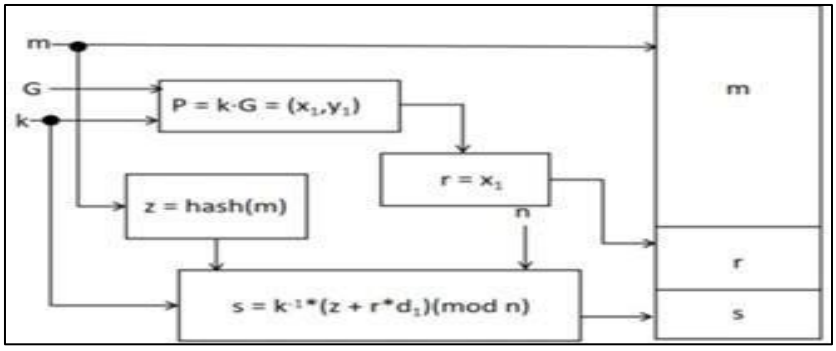

Fig 5 ECDSA Signature

➢ *ECDSA Verification*

To verify signature (r, s) of the message m, verifier should do the following:

- Obtain an authenticated copy of signer's public key Q.
- Verify that r and s are integers in the interval $[1, q - 1]$.
- Compute

$$w = s^{-1} \bmod q \text{ and } H(m) \tag{19}$$

- Compute

$$u_1 = (H(m)w) \bmod q \text{ and } u_2 = (rw) \bmod q \tag{20}$$

- Compute

$$u_1P + u_2Q = (x_0, y_0) \text{ and } v = x_0 \bmod q \tag{21}$$

- Accept the signature if and only if $v = r$.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

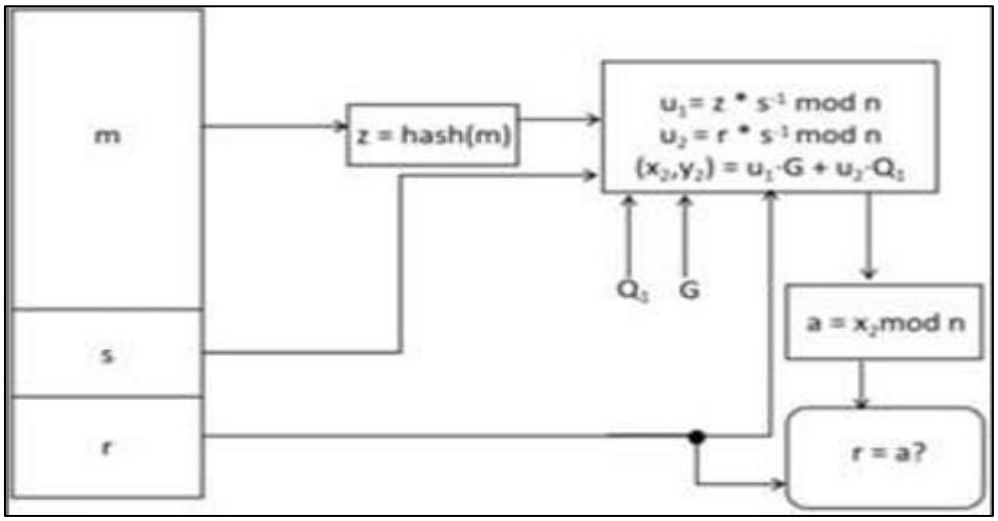

Fig 6 ECDSA Verification

- *Insights of Choosing ECDSA over DSA for Implementing Security to Wireless Sensors to Track Every State Change in the Attack – Graph:*

The basic difference between ECDSA and DSA is in generation of r [4]. The DSA does this by taking the random power α^k mod p and reducing it modulo q, thus obtaining an integer in the interval $[1, q-1]$. (Recall that, in DSA, q is a 160-bit prime divisor of $p-1$, and α is an element of order q in F_p).

The ECDSA generates the integer r in the interval $[1, q-1]$ by taking the $x$coordinate of the random multiple of kP and reducing it modulo q.

To obtain a security level similar to that of DSA, the parameter q should have about 160 bits. If this is the case, then DSA and ECDSA signatures have the same bitlength (320 bits).

Instead of using E and P as system-wide parameters, we could fix only the underlying finite field F_p for all users, and let each user select her own elliptic curve E and point P $E(F_p)$. In this case, the defining equation for E, the coordinates of the point P , and the order q of P must also be included in the user's public key. If the underlying field F_p is fixed, then hardware and software can be built to optimize computations in that field. At the same time, there are an enormous number of choices of elliptic curve E over the fixed field F_p.

The security of ECDSA relies on the Elliptic Curve Discrete Logarithm Problem (ECDLP), which is computationally harder than the integer factorization problem and the discrete logarithm problem used in DSA. Consequently, ECDSA achieves the same security level using significantly smaller key sizes.

Table 2 Equivalent Security Levels of DSA and ECDSA

DSA Key Size (bits)	Equivalent ECDSA Key Size (bits)
1024	160
2048	224
3072	256
7680	384
15360	521

A 256-bit ECDSA key provides security comparable to a 3072-bit DSA key. The reduction in key size directly decreases memory consumption, packet size, and communication overhead.

➢ *Lower Communication Overhead*

Wireless sensors in the proposed system periodically transmit signed status messages to the central controller:

$$Packet = \{SensorID, Timestamp, EventType, Location, State\}$$

The sensor generates the signature

$$S = Sign_{priv}(Hash(Packet))$$

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

And transmits

$$(Packet, S)$$

Because ECDSA signatures are considerably smaller than DSA signatures, fewer bits must be transmitted over the wireless medium. The transmission energy consumed by a wireless sensor can be approximated by

$$Etx = Ptx \times Ttx$$

Where P_{tx} denotes transmission power and T_{tx} denotes transmission time. Since ECDSA produces smaller signatures,

$$TtxECDSA < TtxDSA$$

Which implies

$$E_{tx}ECDSA < EDSA_{tx}$$

This reduction is critical for battery-powered sensors deployed in remote locations.

➢ *Reduced Authentication Latency*

The objective of the proposed surveillance architecture is to minimize interception latency. Before an event can update the AO* graph, the central controller must verify the sensor signature. The total response delay is

$$Tresponse = Tsense + Tsign + Ttransmit + Tverify + TAO* + Tdispatch$$

Where

- T_{sense} is the sensing delay,
- T_{sign} is the signature generation time,
- $T_{transmit}$ is the communication delay,
- T_{verify} is the signature verification time,
- T_{AO*} is the path optimization time,
- $T_{dispatch}$ is the interceptor deployment delay.

The smaller signature size of ECDSA significantly reduces $T_{transmit}$, thereby reducing the overall interception latency.

➢ *Scalability in Large Sensor Networks*

A modern high-security facility may contain thousands of wireless sensors monitoring corridors, entrances, vaults, laboratories, and restricted areas. Assuming N sensors each produce M signed events per minute, the communication load becomes.

$$Traffic = N \times M \times SignatureSize$$

Since

$$SignatureSize_{ECDSA} < SignatureSize_{DSA},$$

The communication bandwidth requirement is substantially reduced, enabling the surveillance framework to scale efficiently. Therefore, ECDSA is considerably more suitable for embedded sensor hardware than traditional DSA.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

CHAPTER EIGHT
IMPROVED ECDSA SCHEME

➢ *SHA-1*

In cryptography, SHA-1 (Secure Hash Algorithm -1) is a hash function which takes an input and produces a 160 bits (20-byte) hash value known as a message digest – typically rendered as 40 hexadecimal digits [15, 16]. It was designed by the United States National Security Agency, and is a U.S. Federal Information Processing Standard (USFIPS). The algorithm has been cryptographically broken but is still widely used. SHA-1 is based on principles similar to those by Ronald L. Rivest of MIT in the design of MD4 and MD5 message [33] digest algorithms, but generates a larger message digest [34].

➢ *SHA-1 Algorithm*

Require: Message M

Ensure: 160-bit hash value HH

Initialization

$h_0 \leftarrow 0x67452301$

$h_1 \leftarrow 0xEFCDAB89$

$h_2 \leftarrow 0x98BADCFE$

$h_3 \leftarrow 0x10325476$

$h_4 \leftarrow 0xC3D2E1F0$

$ml \leftarrow$ message length in bits

Pre-processing

Append bit 1 to the message.

Append 0 bits until the message length is congruent to 448 (mod 512).

Append the original message length ml as a 64-bit big-endian integer.

Process each 512-bit block

forall *512-bit chunk C of the padded message* **do**

Divide C into sixteen 32-bit words

$$W[0], W[1], \ldots, W[15].$$

for $i = 16$ *to* 79 **do**

$$W[i] \leftarrow \mathrm{ROL}_1(W[i-3] \oplus W[i-8] \oplus W[i-14] \oplus W[i-16])$$

Initialize

$$a \leftarrow h_0, \quad b \leftarrow h_1, \quad c \leftarrow h_2, \quad d \leftarrow h_3, \quad e \leftarrow h_4.$$

for $i = 0$ *to* 79 **do**

if $0 \leq i \leq 19$ **then**

$$f = (b \wedge c) \vee (\neg b \wedge d)$$

$$k = 0x5A827999$$

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

19: $20 \leq i \leq 39$

20:

$$f = b \oplus c \oplus d$$

21:

$$k = 0x6ED9EBA1$$

22: $40 \leq i \leq 59$

23:

$$f = (b \wedge c) \vee (b \wedge d) \vee (c \wedge d)$$

24:

$$k = 0x8F1BBCDC$$

else

25:

$$f = b \oplus c \oplus d$$

26:

$$k = 0xCA62C1D6$$

27:

28:

$$temp \leftarrow ROL_5(a) + f + e + k + W[i]$$

29: $e \leftarrow d$

30: $d \leftarrow c$

31: $c \leftarrow ROL_{30}(b)$

32: $b \leftarrow a$

33: $a \leftarrow temp$

34:

35: Update hash values

36: $h_0 \leftarrow h_0 + a$

37: $h_1 \leftarrow h_1 + b$

38: $h_2 \leftarrow h_2 + c$

39: $h_3 \leftarrow h_3 + d$

40: $h_4 \leftarrow h_4 + e$

41:

42: **Output**

$$HH = (h_0 \ll 128) \mid (h_1 \ll 96) \mid (h_2 \ll 64) \mid (h_3 \ll 32) \mid h_4.$$

43: **return** *HH*

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

➢ *Signature Generation*

When Alice sends the message to Bob, and then obtains a digital signature (r, s) generated by following steps.

- Select a random k in [1, n − 1].
- Calculate a curve point $kG = (x_1, y_1)$.
- Calculate $r = x_1 \bmod n$. If $r = 0$, then go to 1st step.
- Calculate $e = SHA - 1(m)$.
- Calculate $s = (e + k + rd) \bmod n$. If $s = 0$, then go back to 1st step.
- Send the message m and digital signature (r, s).

➢ *Signature Verification*

From the steps of digital signature generation, we can design the key step of verification by Backward Recurrence Algorithm:

$$X = kG \tag{22}$$

$$= (s - e - rd)G \tag{23}$$

$$= (s - e)G - rdG \tag{24}$$

$$= (s - e)G - rQ \tag{25}$$

$$= (x_1, y_1) \tag{26}$$

Bob should do the followings to verify the digital signature:

- Verify that r and s are integers in [1, n − 1]. If not, the signature is invalid.
- Calculate $e = SHA - 1(m)$.
- Calculate $w = (s - e) \bmod n$.
- Calculate a curve $X = wG - rQ = (x_1, y_1)$.
- If $X = O$, the signature is invalid and refuse to it, else calculate $v = x_1 \bmod n$.
- Bob will accept the digital signature if and only if $v = r$.

From the verification steps above, we can see the digital signature verification is the reverse of digital signature generation.

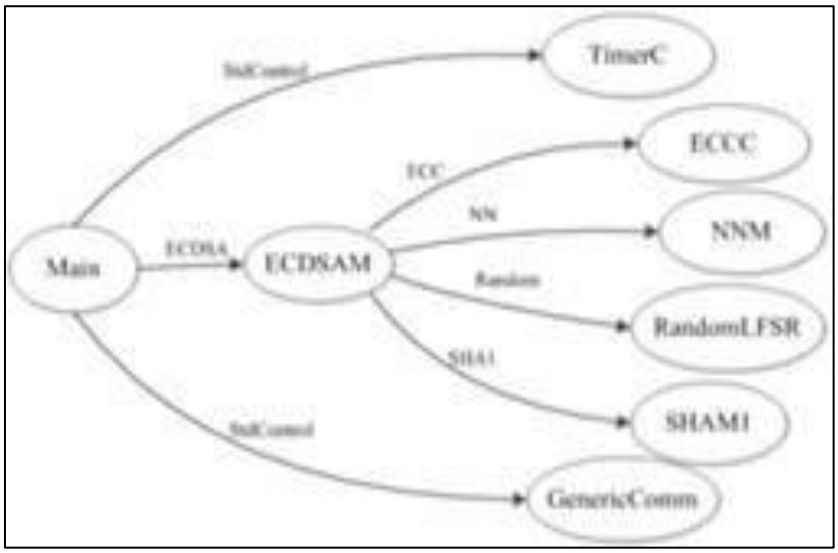

Fig 7 ECDSA Verification

➢ *Cost Analysis*

Sensor nodes are extremely limited in terms of energy supply, storage space and other resources. We suppose that the consuming time of plus operation, multiply operation, modular arithmetic and Hash operation were E_1, E_2, E_3 *and* E_4 respectively. Assuming the cost of ellipse curve initialization is E, therefore, the cost comparison of the improved ECDSA with the original ECDSA listed in Table 3 [16].

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

Table 3 Cost Comparison of Improved ECDSA and Original ECDSA

Schemes	Original ECDSA	Improved ECDSA
Generation	$E + n(E_1 + 3E_2 + 2E_3 + E_4)$	$E + n(2E_1 + 2E_2 + E_3 + E_4)$
Verification	$E + n(E_1 + 4E_2 + 3E_3 + E_4)$	$E + n(2E_1 + 3E_2 + 2E_3 + E_4)$

Elliptic curves system over finite fields, which is very time-consuming. Inverse operation is much slower than multiplication. Knuth [20] pointed that using Extended Euclidean algorithm to achieve modular inverse also need to complete the $0.843 \log_2 n + 1.47$ times division operation averagely. The improved signature scheme utilize plus operation instead of using time-consuming modular inverse, which can effectively reduce the computational complexity.

➢ *Security Analysis*

The difficulties associated with the attacks are based on the solution of the ECDLP, and the security resulted from such problems is still sufficient under the reasonable computational complexity.

- If an attacker got the public key Q, who wants to obtain its private key d through Q. This is one problem for solving the elliptic curve discrete logarithm, so this approach is not feasible.
- If an attacker obtained $m,(r, s)$, who wants to obtain the private key d by solving

$s = (SHA - 1(m) + k + rd) \bmod n$ (8.6) $d = ((s - (SHA - 1(m)) - k)r - 1 \bmod n$ (27)

Because k is selected by the signer randomly, so this approach is also not feasible.

- If the attacker got $m,(r, s)$, although the attacker cannot acquire the private key d of the signer, but want to fake signature by the equation $s = (e + k + rd) \bmod n$. Only have to generate k_1 randomly, to find s_1 by r_1. When authentication only by the equation $X = (s_1 - e - r_1 d)G \bmod n = k_1 G /= (x_1, y_1)$. The attacker even though can avoid solving d, however, because of random k, so forgery is not feasible.

➢ *Efficiency Analysis*

- Implement inverse-free operation. In the usual elliptic curve encryption or signature process, inverse operation is the main computational burden. The improved scheme eliminates the inverse operation, and the experimental results show that the above scheme can correctly judge the validity of the signature, and ease the required computing. Compared to other elliptic curve signature schemes, the system reduce time consumption and improve the efficiency of signature.
- By avoiding twice modular inversion, which can effectively reduce the generation time of a digital signature and digital signature verification time. By the following average time in the first nine rounds can draw a conclusion that digital signature generation time is reduced by 7.3%, the digital signature verification time reduced by 6.1%.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

CHAPTER NINE
AO* ALGORITHM

➢ *Introduction to AO* Procedure*

AO* Algorithm is an informed search and works as best first search. AO* Algorithm is based on problem decomposition (Breakdown problem into small pieces).The algorithm does not explore all the solution path once it find a solution. It's an efficient method to explore a solution path. AO* is often used for the common pathfinding problem in applications such as video games, but was originally designed as a general graph traversal algorithm. It finds applications in diverse problems, including the problem of parsing using stochastic grammars in NLP. Other cases include an Informational search with online learning. It is useful for searching game trees, problem solving etc.

➢ *AO* Procedure*

Procedure AO* Begin

- Given the goal node INIT in the graph G; evaluate h' at INIT;
- Repeat

✓ Trace the marked arcs from the node INIT, if any such exists, and select one of the unexpanded nodes, named NODE, that occurs on this path, for expansion.
✓ If NODE cannot be expanded, Then assign FUTILITY as the h' value of NODE, indicating that NODE is not solvable;

Else for each such successor, called SUCCESSOR, which is not an ancestor of NODE, do Begin

✓ Append SUCCESSOR to the graph G.
✓ If SUCCESSOR is a terminal node Then label it SOLVED and set its h' value 0.
✓ If SUCCESSOR is not a terminal node Then estimate its h' value;

End;

($_c$) Initialize S to NODE; ($_d$) Repeat

- Select from S a node, none of whose descendants belong to S. Call it CURRENT and remove it from S.
- Estimate the cost of each of the arcs, emerging from CURRENT. The cost of each arc is equal to the sum of h' value of each of the nodes at the end of the arc plus the cost of the arc itself. The new h' value of CURRENT is the minimum of the cost just computed for the arcs emerging from it.
- Label the best path out of CURRENT by marking the arc that had the least cost as computed in the last step.
- If all of the nodes connected to CURRENT through the new marked arcs have been labeled SOLVED, Then mark the CURRENT SOLVED.
- If CURRENT is marked SOLVED or the cost of CURRENT was changed, Then propagate its new status back up the tree, add all the ancestors of CURRENT to S.

Until S is empty.

Until INIT is labeled solved or its h' value becomes greater than a maximum level called FUTILITY:

End.

➢ *Why AO*?*

It represents an AND-OR graph algorithm that is used to find more than one solution by ANDing more than one branch. In this algorithm we follow a similar procedure but there are constraints traversing specific paths. When you traverse those paths, cost of all the paths which originate from the preceding node are added till that level, where you find the goal state regardless of the fact whether they take you to the goal state or not. Security environments are dynamic. *AO* allows security systems to re-evaluate and adjust their remediation strategies in real-time if an attacker attempts to bypass a firewall or alters their attack vector.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

➢ *AO* Interception Optimization*

Algorithm 2: AO* Interception Optimization

```
Initialize graph G = (V, E);
Assign heuristic h(v);
Set C(v) = h(v);
while solution not found do
    Select node minimizing f(v) = g(v) + h(v);
    Expand node;
    for each child u do
        if φ(v) = OR then
            C(v) = min[c(v, u) + C(u)];
        else
            C(v) = Σ[c(v, u) + C(u)];
    Backpropagate;
```

➢ *Complexity of AO* Interception Optimization Algorithm*

- *Time Complexity*

In the worst-case scenario, the algorithm can be exponential: $O(b^d)$ where b is the branching factor and d is the depth. However, a highly accurate heuristic $h(v)$ drastically prunes the search space, reducing effective runtime.

- *Space Complexity*

$O(b^d)$ because *AO* must keep the explicit search graph in memory to perform the backpropagation step.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

CHAPTER TEN
MATHEMATICAL FRAMEWORK

To mathematically prove that the proposed system reduces the Time-To-Interception(TTI) while maintaining cryptographic integrity, we must formalize the environment as a Spatio-Logical State Space. Formal System Definition:

➢ *Formal System Definition*

We model the security environment as a spatio-logical system:

$$S = (G, A, I, \Sigma) \tag{28}$$

$G = (V, E, \gamma)$: AND-OR graph facility

A : *Attacker Dynamics*

I : *Interceptor Dynamics*

Σ : *Cryptographic V alidation Constarints*

G captures the logical dependencies. A, I captures continuous-time motion. Σ ensures the data integrity, preventing adversarial corruption.

Define a function φ such that

$$\varphi : V \rightarrow \{\text{AND, OR, LEAF, GOAL}\} \tag{29}$$

Now defining the logical satisfaction function L such that,

$$L(v) = \begin{cases} \bigwedge_{u \in children(v)} L(u), & \text{if } \varphi(v) = \text{AND}, \\ \bigvee_{u \in children(v)} L(u), & \text{if } \varphi(v) = \text{OR}, \\ \cdot\ \gamma(v), & \text{if } \varphi(v) = \text{LEAF}, \\ \cdot\ 1, & \text{if } \varphi(v) = \text{GOAL}. \end{cases}$$

$$\varphi(v) = AND \rightarrow \textit{all subgoals required},$$
$$\varphi(v) = OR \rightarrow \textit{alternative paths}$$

➢ *Cost Function*

Now we define the recursive cost function for obtaining the optimal path,

$$C(v) = \begin{cases} 0, & \phi(v) = GOAL \\ \cdot\ \min(c(v,u) + C(u)), & \phi(v) = OR \\ \cdot\ \sum_{u}^{u} (c(v,u) + C(u)), & \phi(v) = AND \end{cases} \tag{30}$$

Defining edge cost for attacker:

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

$$c(v, u) = \frac{d(v, u)}{v_A} + s(u) \tag{31}$$

Where $d(v, u)$ = length of the shortest spatial path between v *and* u .

$s(u)$ = time required for the attacker to satisfy or bypass the security condition at u.

v_A = rate at which the attacker moves through physical space

Therefore, total attacker time, $\tau(v) = C(v)$ as $C(v)$ aggregates costs using φcontrolled recursion

$$\boxed{\tau(v) = C(v)} \tag{32}$$

Now defining the interceptor time,

$$\delta(v) = \min_{\pi_I} \Sigma \frac{d(v, w)}{v_I} \tag{33}$$

Where π_I = interceptor path = $(v_0, v_1, \ldots, v_k)$

v_I = speed of the interceptor

Therefore, it is quite evident that, Interception Condition is given by,

$$\boxed{\delta(v) \leq \tau(v) \text{ or, } \delta(v) \leq C(v)} \tag{34}$$

Since $\tau(v) = C(v)$.

- *Heuristic Formation*

  Defining the heuristic, $h(v)$,

$h(v)$ = *estimated remaining cost* (*time*) *for the attacker to reach the goal from node* $v = ad(v, \text{GOAL}) + \beta s(v) + \sigma p(v)$

Where, $d(v, GOAL)$ = remaining physical distance $s(v)$ = estimated remaining security delay $\rho(v)$ = interceptor influence (risk/pressure near the node) α, β, σ = weighting constants

- *AO* Evaluation*

  Now the AO* algorithm uses the cost function

$f(v) = g(v) + h(v)$ (35)

$g(v)$ = cost already incurred $h(v)$ = prediction of future cost

It can be easy to show that the above mentioned heuristic function $h(v)$ is admissible. And,

$$h(v) \leq C^*(v) \tag{36}$$

AO* Update Rule:

$$C'(v) = \begin{cases} \min_{u \in children(v)} [c(v, u) + C'(u)], & \text{if } \varphi(v) = OR \\ \Sigma_{u \in children(v)} [c(v, u) + C'(u)], & \text{if } \varphi(v) = AND \end{cases} \tag{37}$$

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

Cryptographic Constraints:

$$E^{*}=\{(v,u) \in E : Verify(Sign) = 1\}$$

Then all members of E ∗ are said to be valid edges.

Modified Edge Cost:

$$c^{*}(v,u) = \begin{cases} d(v,u), & \text{if } (v,u) \in E^{*} \\ \infty, & otherwise \end{cases} \quad (38)$$

Therefore, new cost recursion

$$C^{*}(v) = \begin{cases} \min[c^{*} + C^{*}], & \text{if } \varphi(v) = OR \\ \Sigma[c^{*} + C^{*}], & \text{if } \varphi(v) = AND \end{cases} \quad (39)$$

➢ *Existence of Minimum Attacker Time and Validity of Graph Updates*

Lemma 1. *For any node v, C*(*v*) *computed using φ is the minimum attacker time to satisfy all logical constraints of v. That means we have to show, τ* (*v*) = *C*(*v*).

- *Proof:*

Case – 1: *φ*(*v*) = *GOAL*

If *φ*(*v*) = *GOAL*, then by definition, *C*(*v*) = 0.

So after approaching the goal, no further cost is required. Therefore, *τ* (*v*) = 0 = *C*(*v*).

Case – 2: *φ*(*v*) = *LEAF*

From the definition of the logical satisfaction function, *C*(*v*) = *γ*(*v*).

Since *γ*(*v*) encodes the sensor-trigger/anomaly cost, it also represents the minimal effort (or time) required to activate this state. Therefore, *τ* (*v*) = *γ*(*v*) = *C*(*v*).

Proving the theorem for *φ*(*v*) = *OR* and *φ*(*v*) = *AND* requires the Principle of Mathematical Induction.

The base cases have already been established in Case–1 and Case–2. We now proceed to the inductive hypothesis.

- *Inductive Hypothesis:*

Assume that for every child *u* of node *v*, *C*(*u*) = *τ* (*u*).

We prove for node *v*.

- *Induction Step:*

Case – 3: *φ*(*v*) = *OR*

$$C(v) = \min_{u \in children(v)} [c(v,u) + C(u)]$$

Then, by the inductive hypothesis,

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

$$C(v) = \min_{u} [c(v, u) + \tau(u)] \tag{1}$$

The attacker chooses one child path, and the total time is equal to the transition time plus the future cost.

Therefore,

$$\tau(v) = \min_{u} [t_A(v \rightarrow u) + \tau(u)] .$$

But,

$$c(v, u) = t_A(v \rightarrow u).$$

Hence,

$$\tau(v) = \min_{u} [c(v, u) + \tau(u)] . \tag{2}$$

By comparing (1) and (2), we obtain

$$C(v) = \tau(v).$$

Case – 4: φ(v) = AND

$$C(v) = \sum_{u \in children(v)} [c(v, u) + C(u)]$$

Then, by the inductive hypothesis,

$$C(v) = \sum_{u} [c(v, u) + \tau(u)] \tag{3}$$

The attacker must satisfy all sub-conditions, so the total time is the sum of all required actions.

We know,

$$L(v) = \bigwedge_{u} L(u).$$

Therefore,

$$\tau(v) = \sum_{u} [t_A(v \rightarrow u) + \tau(u)] .$$

But,

$$c(v, u) = t_A(v \rightarrow u).$$

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

Hence,

$$\tau(v) = \sum_{u} [c(v,u) + \tau(u)]. \tag{4}$$

Therefore, by comparing (3) and (4), we obtain

$$C(v) = \tau(v).$$

Hence, the lemma is proved.

Theorem 1. *Using AO* graph on G defined by φ, If* $\delta(v)$ = *interceptor travel time and* $\tau(v)$ = *attacker arrival time. Then,*

$$\exists\, v_* \in V \text{ such that } \delta(v_*) \leq \tau(v_*)$$

And therefore,

$$TTIAO* < TTIReactive$$

Provided that, AO uses an admissible heuristic and graph updates are valid (ECDSA ensures correctness).*

Proof. From the above lemma,

$$C(v) = \tau(v)$$

Since the heuristic is admissible,

$$h(v) \leq C(v).$$

AO* guarantees,

$$C(v) = C^*(v).$$

Now as per above defined, interceptor time,

$$\delta(v) = \min_{\pi_i} \sum \frac{d(v,w)}{v_i}.$$

It is quite evident that,

$$\delta(v) \geq 0, \qquad \forall\, v.$$

Let us define, Interception Gap as,

$$\Delta(v) = \tau(v) - \delta(v)$$

Now if,

- $\Delta(v) > 0$: attacker arrives first.
- $\Delta(v) = 0$: Simultaneous arrival.
- $\Delta(v) < 0$: Interceptor arrives first.

Since V is finite, then by well-ordering principle,

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

$$\exists\, v^* \in V \text{ such that } \Delta(v^*) = \min_{v \in V} \Delta(v)$$

Case – 1: If $\Delta(v^*) \leq 0$, then

$$\delta(v^*) \leq \tau(v^*)$$

Case – 2: If $\Delta(v^*) > 0$, then

$$\delta(v) > \tau(v) \qquad \forall v$$

Now we will show Case–2 is a contradiction.

Let, if possible, Case–2 holds i.e.

$$\Delta(v) > 0 \qquad \forall v$$

Consider the goal node v_g,

$$\tau(v_g) = 0$$

$$\delta(v_g) > 0$$

So,

$$\Delta(v_g) = \tau(v_g) - \delta(v_g) = 0 - \delta(v_g) = -\delta(v_g) < 0$$

The above is a contradiction to Case–2.

Therefore,

$$\exists\, v^* \in V \text{ such that } \Delta(v^*) \leq 0$$

In case of Reactive System, Interceptor moves to current attacker position v_c; that means,

$$\delta(v_c) > \tau(v_c)$$

As attacker already arrived.

Therefore, AO* predicts optimal node v^* such that

$$\delta(v^*) \leq \tau(v^*)$$

Hence,

$$TTI_{AO*} < TTI_{\text{Reactive}} \qquad \textbf{(Proved).}$$

- *Final Optimization*

  Optimal interception node, $v^* = \arg\min_{v \in V} \Delta(v)$

Then Final Optimization Objective Function is,

$$\min_{\pi_I} \max_{\pi_A} \min_{v \in V} [C(v; \varphi, \Sigma) - \delta(v)] \qquad (40)$$

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

Where all the mentioned symbols in final optimization objective function are previously described. This is an unconstrained optimization problem.

In the objective function, we have used worst-case attacker strategy respect to interceptor and best interceptor deployment.

- *Final Decision Rule*

Deploy interceptor at v^* such that $\delta(v^*) \leq \tau(v^*)$.

➢ *Final Optimization Algorithm*
Procedure CVPAO* (Cryptographically Verified Predictive AO*) Begin

- Given the root node INIT in the graph G, the validation constraints Σ, attacker dynamics I_A, interceptor dynamics I_I, and interceptor initial position I_0; evaluate C(INIT) ← h(INIT) and set status(INIT) ← UNSOLVED;
- Perform cryptographic validation for all edges in parallel:

For every edge $(u, v) \in E$ do

Begin

✓ If VerifyECDSA(u, v) = TRUE, Then set $c(u, v) \leftarrow c(u, v)$; ($_b$) Else set $c(u, v) \leftarrow \infty$;

End;

- *Repeat*

✓ Trace the marked arcs from the node INIT, if any such exists, and select one of the unexpanded nodes, named NODE, that occurs on this path, for expansion.
✓ If NODE cannot be expanded, Then assign FUTILITY as the C value of NODE, indicating that NODE is not solvable;

Else for each such successor, called SUCCESSOR, which is not an ancestor of NODE, do Begin

✓ Append SUCCESSOR to the graph G.
✓ If φ(SUCCESSOR) = GOAL Then label it SOLVED and set C(SUCCESSOR) ← 0.
✓ If φ(SUCCESSOR) = LEAF Then set C(SUCCESSOR) ← γ(SUCCESSOR).
✓ If SUCCESSOR is neither a GOAL nor a LEAF Then estimate its
C value using the admissible heuristic h(SUCCESSOR);

End;

✓ Initialize S to NODE; ($_d$) Repeat
✓ Select from S a node, none of whose descendants belong to S. Call it CURRENT and remove it from S.

✓ Estimate the cost of each of the hyperarcs emerging from CURRENT.

If φ(CURRENT) = OR Then the cost is

$$\min_{u \in \text{children(CURRENT)}} [c^*(\text{CURRENT}, u) + C(u)] \tag{41}$$

If φ(CURRENT) = AND Then the cost is

$$\sum_{u \in \text{children(CURRENT)}} [c^*(\text{CURRENT}, u) + C(u)] \tag{42}$$

Set the new C(CURRENT) value to the minimum cost just computed.

✓ Label the best path out of CURRENT by marking the arc (or set of arcs for AND branches) that had the least cost as computed in the last step.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

- ✓ If all of the nodes connected to CURRENT through the new marked arcs have been labeled SOLVED, Then mark the CURRENT SOLVED.
- ✓ If CURRENT is marked SOLVED or the cost of CURRENT was changed, Then propagate its new status back up the tree, add all the ancestors of CURRENT to S.

Until S is empty.

Until INIT is labeled SOLVED or its C(INIT) value becomes greater than FUTILITY;

- Compute Attacker Arrival Time:

For every node $v \in V$ do set $\tau(v) \leftarrow C(v)$;

- Compute Interceptor Travel Time:

For every node $v \in V$ **do** compute $\delta(v) \leftarrow \min_{\pi_I} \sum_{w \in \pi_I} \frac{d(w)}{v_I}$;

- Compute Interception Gap:

For every node $v \in V$ do compute $\Delta(v) \leftarrow \tau(v) - \delta(v)$;

- Bottleneck Optimization to select candidate interception node v^*:

Identify CandidateSet $\leftarrow \{v \in V : \delta(v) \leq \tau(v)\}$;

If CandidateSet $\neq \emptyset$ Then $v^* \leftarrow \arg\min_{v \in \text{CandidateSet}} \Delta(v)$;

Else $v^* \leftarrow \arg\min_{v \in V} \Delta(v)$;

- Minimax Optimization:
  Determine the optimal interceptor deployment policy:

$$\pi_I^* \leftarrow \arg\min_{\pi_I} \max_{\pi_A} \min_{v \in V} [C(v; \varphi, \Sigma) - \delta(v)] \quad (43)$$

- Deploy interceptor to v^* and monitor incoming signed state updates;
- Repeat

- ✓ Monitor communication channels for incoming authenticated events.
- ✓ If new authenticated event arrives Then Begin
- ✓ Update graph G.
- ✓ Recompute C values, v^*, and policy π^*_I by restarting from Step 2.
- ✓ Redeploy interceptor.

End;

Until Goal neutralized OR attack terminated.

End.

➢ *Time Complexity and Space Complexity of Final Optimization Algorithm*
To analyze the performance of the Cryptographically Verified Predictive AO* (CVPAO*) algorithm, let:

- $|V|$ be the total number of nodes in the AND-OR attack graph.
- $|E|$ be the total number of edges in the graph.
- b be the maximum branching factor of the graph.
- d be the maximum depth of the graph structure.
- T_{verify} be the processing time required to validate a single ECDSA signature.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

- *Time Complexity*
  The overall time complexity is derived from the sequential execution of its core algorithmic phases:

✓ Initialization & Cryptographic Validation (Steps 1–2): Initializing values across all nodes takes linear time $O(|V|)$. The cryptographic phase processes every edge exactly once to verify authenticity, which scales bounded by edge size and cryptographic verification time:

$$T\text{crypto} = O(|E| \cdot T\text{verify}) \quad (44)$$

✓ Predictive Search & Cost Propagation (Step 3): In the worst-case scenario (uninformative heuristic), the AO* sub-routine expands up to all nodes. Processing AND/OR logic constraints across child nodes requires $O(b)$ steps per expansion. When node costs change, ancestor propagation via the tracking set S can scale up to the depth of the graph, yielding $O(d \cdot b)$.
Accumulated across all nodes, the search and back-propagation phase requires:

$$T_{\text{search}} = O(|V|^2 \cdot b) \quad (45)$$

✓ Dynamics & Interception Evaluation (Steps 4–6): Computing the attacker arrival time $\tau(v)$ and gap matrix $\Delta(v)$ requires linear passes $O(|V|)$. Determining interceptor travel times $\delta(v)$ map matches a shortest-path evaluation (e.g., Dijkstra's algorithm) over the layout matrix, taking $O(|E| + |V| \log |V|)$.

✓ Optimization & Minimax Evaluation (Steps 7–8): The bottleneck selection runs in linear scan time $O(|V|)$. The minimax formulation evaluates a zero-sum sequence over the paths π_I and π_A. Traversing this execution space across a max-depth path yields an exponential game-tree complexity:

$$T_{\text{minimax}} = O(|V| \cdot b^d) \quad (46)$$

- *Total Worst-Case Time Complexity:*
  Summing all deterministic components for a static execution pass yields:

$$O\left(|E| \cdot T_{\text{verify}} + |V|^2 \cdot b + |V| \cdot b^d\right) \quad (47)$$

Note: If K authenticated dynamic environmental event triggers occur in the runtime tracking loop, the algorithm re-computes, multiplying the static execution runtime by a factor of K.

- *Space Complexity*
  The storage footprint of the algorithm tracks network structures, tracking spaces, and optimization arrays:

✓ Graph Topology State: Storing the modified attack tree $G = (V, E)$ along with verified weights $c(u, v)$ and logical flags φ requires $O(|V| + |E|)$ space using adjacency lists.
✓ Evaluation Metrics Tables: Flat operational arrays maintaining system metrics (τ, δ, Δ, and status tracking) require strict linear memory: $O(|V|)$.
✓ Search Stack Space: The maximum allocation footprint of the closed/open subsets and the ancestral propagation tracker S is upper-bounded by $O(|V|)$. The recursive execution context stack for the minimax game tree scales relative to depth: $O(d)$.

- *Total Space Complexity:*
  Because the network topologies dominate linear structural parameters ($|V| + |E| \gg d$) , the complete space bound simplifies cleanly to:

$$O(|V| + |E|) \quad (48)$$

- *Analytical Breakdown of CVPAO* Complexities*
  The following sections provide the mathematical justification, graph-theoretic proofs, and operational limits of the Cryptographically Verified Predictive AO* (CVPAO*) algorithm.

✓ *Deep-Dive Time Complexity Analysis*
The overall worst-case time complexity of $O\left(|E| \cdot T_{\text{verify}} + |V|^2 \cdot b + |V| \cdot b^d\right)$ is dictated by three primary computational bottlenecks.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

- *The Cryptographic Validation Bottleneck*

In conventional pathfinding algorithms, edge traversal costs $c(u, v)$ are assumed to be static, readily accessible constants. In CVPAO*, path searching is coupled directly with real-time cryptographic validation.

✓ Mechanism: Prior to structural evaluation, every edge candidate must be cryptographically validated utilizing Elliptic Curve Digital Signature Algorithm (ECDSA) constraints.
✓ Mathematical Analysis: ECDSA signature verification relies on computationally intensive point multiplication operations over elliptic curves in finite fields. If T_{verify} is the execution time for a single validation, the verification of all transition possibilities scales linearly with the total edge count:

$$T_{\text{crypto}} = \sum_{e \in E} T_{\text{verify}} = O(|E| \cdot T_{\text{verify}}) \tag{49}$$

- *The AO* Search and Cost Propagation Bottleneck*

Because the threat graph G contains AND-OR logical relationships (where some nodes require all child states to succeed, while others require only one), standard sequential search algorithms like A* cannot be applied. Instead, a hypergraph-based AO^* routine is required.

✓ **Analytical Modeling:** The search complexity is driven by two alternating operations: top-down node expansion and bottom-up cost propagation.
✓ **Quadratic Worst-Case Scenario:** In a poorly balanced, highly recursive, or highly dense graph structure, updating a single leaf node's cost can trigger a cascade of cost recalculations propagating all the way up to the root node. In the absolute worst case, a single expansion forces an update across almost all previously expanded states in the graph.
✓ Summing this behavior over $|V|$ potential node expansions yields:

$$T_{\text{search}} \approx \sum_{i=1}^{|V|} (\text{Cost of Expansion at step } i + \text{Cost of Propagation}) \tag{50}$$

$$T_{\text{search}} \approx \sum_{i=1}^{|V|} (b + i \cdot b) \approx O(|V|^2 \cdot b) \tag{51}$$

- *The Minimax Game-Theoretic Bottleneck*

Step 8 evaluates the minimax equilibrium formulation:

$$\pi_I^* = \arg\min_{\pi_I} \max_{\pi_A} \min_{v \in V} [C(v; \varphi, \Sigma) - \delta(v)] \tag{52}$$

This models a zero-sum, multi-agent game between the attacker choosing optimal path policy π_A and the defender choosing interception policy π_I.

Evaluating this equilibrium requires traversing the complete game-tree trajectory space up to the maximum depth d. For a maximum branching factor b, the state-space tree grows exponentially. Because this game projection must be evaluated across candidate nodes, the worst-case evaluation complexity scales as:

$$T_{\text{minimax}} = O(|V| \cdot b^d) \tag{53}$$

While practical heuristics and alpha-beta pruning techniques can prune non-optimal search branches dynamically, the theoretical upper bound remains strictly exponential.

➢ *Deep-Dive Space Complexity Analysis*

The space complexity is bounded at a highly efficient linear rate of $O(|V|+|E|)$. This allocation footprint is justified by analyzing structural memory layout components.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

- *Graph Topology Representation*
  Storing the AND-OR graph $G = (V, E)$ requires:

✓ An adjacency list to track node connections, consuming $O(|V|+|E|)$ memory.
✓ Flat primitive variables stored per node:

Type labels $\varphi(v) \in \{\text{AND, OR, LEAF, GOAL}\}$, status markers $\text{status}(v) \in \{\text{SOLVED, UNSOLVED}\}$, and numeric evaluation arrays ($\tau(v)$, $\delta(v)$, $\Delta(v)$). These consume exactly $O(|V|)$ space.

- *Search State Space Tracking*

✓ OPEN and CLOSED Lists: Unlike tree-search equivalents that might store multiple duplicate path representations, graph-based *AO* enforces unique node references. A node exists in either the OPEN or CLOSED set, never both, limiting tracking space to $O(|V|)$.
✓ The Ancestor Set (S): The tracking set S propagates cost changes bottomup. In the worst-case configuration (a single degenerate path), S contains at most d elements (where $d \leq |V|$). Thus, S requires at most $O(|V|)$ space.

- *Recursive Minimax Execution Stack*

The game-theoretic policy selection is evaluated using depth-first search (DFS) recursion. Since DFS only requires storing the active path from the root to the leaf, the execution stack depth is directly proportional to the maximum height of the decision tree, requiring $O(d)$ space.

Given that the structural network components dominate both linear tracking parameters and tree depth ($|V| + |E| \gg d$), the complete space bound simplifies cleanly to:

$$\text{Space} = O(|V| + |E| + d) \Longrightarrow O(|V| + |E|) \tag{54}$$

➢ *Complexity Profile Summary*
The operational scaling properties of the individual CVPAO* components are summarized below:

Table 4 CVPAO* Complexity Performance Profile

Algorithmic Phase	Key Mathematical Driver	Complexity Bound
Cryptographic Validation	Signature verification overhead per edge	$O(\|E\| \cdot T_{\text{verify}})$
Search Graph Generation	Quadratic bottom-up recursive update cascade	$O(\|V\|^2 \cdot b)$
Minimax Policy Selection	Exponential game-tree decision-space expansion	$O(\|V\| \cdot b^d)$
Data Structure Allocation	Physical storage of nodes, edges, and flat arrays	$O(\|V\| + \|E\|)$

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

CHAPTER ELEVEN
CONCLUSIONS

➢ *Achievement of this Formulation*

- *Logical Correctness (via φ)*

✓ Captures AND/OR dependencies
✓ Models real attack sequences

- *Optimal Prediction (via AO*)*

✓ Computes minimal attacker time
✓ Uses admissible heuristic

- *Security Guarantee (via Σ)*

✓ Ensures only valid transitions
✓ Prevents adversarial manipulation

- *Physical Interception (via δ(v))*

✓ Converts spatial movement into time
✓ Enables real-world deployment

➢ *Achievement of the Final Optimization Algorithm*

The Cryptographically Verified Predictive AO* (CVPAO*) algorithm introduces a crucial paradigm shift in autonomous network defense, state-space search, and threat-response mitigation. By unifying zero-trust cryptographic verification, predictive temporal projection, and game-theoretic optimization, CVPAO* overcomes fundamental security bottlenecks that standard pathfinders cannot address.

- *Core Technological Innovations & Achievements*

✓ *Trust-Aware Pathfinding (Zero-Trust State Space)*

Conventional state-space pathfinders (such as Dijkstra, A*, or standard AO*) implicitly assume that the underlying graph topology $G = (V, E)$ and edge weights $c(u, v)$ are correct and uncompromised. In adversarial environments, an attacker can hijack network resources or spoof state updates to mislead routing calculations.

- The Achievement: CVPAO* natively couples cryptographic validation (ECDSA signature validation) directly into the edge relaxation and expansion cycle.
- Mathematical Impact: The algorithm prunes unauthenticated transitions prior to evaluation, restricting the solvable search space strictly to mathematically proven, secure edges:

$$c^*(u, v) = \begin{cases} c(u, v) & \text{if VerifyECDSA}(u, v) = \text{TRUE} \\ \infty & \text{otherwise} \end{cases} \quad (55)$$

✓ *Modeling Multi-Stage Attacks via Logical Hypergraphs*

Modern cyber threats, such as Advanced Persistent Threats (APTs), rarely progress along linear paths. They leverage parallel vectors that correspond to AND-OR structures (e.g., compromising *both* database A and system backup B, or *either* email server C or gateway D).

- The Achievement: Rather than converting complex attack structures into simplified directed graphs, CVPAO* computes optimal policies natively over logical hypergraphs.
- Mathematical Impact: It provides accurate cumulative risk estimates, modeling branching parallel and serial pathways simultaneously to compute the exact upper bound of attacker arrival times $\tau(v)$.

✓ *Proactive Defense via the Interception Gap Metric*

Standard Intrusion Detection Systems (IDS) are fundamentally reactive, generating alerts only after a node is already compromised.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

- The Achievement: The algorithm introduces the Interception Gap ($\Delta(v)$), which models the temporal race condition between the attacker's trajectory $\tau(v)$ and the interceptor's deployment delay $\delta(v)$.
- Mathematical Impact: The delta computation enables proactive threat mitigation, allowing defensive controls (such as dynamic micro-segmentation, software-defined network isolation, or honeytoken activation) to be deployed at bottleneck nodes *before* the attacker reaches them:

$$\Delta(v) = \tau(v) - \delta(v) \quad \text{where } \delta(v) = \min_{\pi_I} \sum_{w \in \pi_I} \frac{d(w)}{v_I} \tag{56}$$

✓ *Provably Optimal Game-Theoretic Defense*

Adaptive adversaries dynamically change their strategies in response to defending countermeasures. Static firewall rules or fixed routing policies are easily bypassed once mapped by an attacker.

- The Achievement: Step 8 implements a **Minimax optimization framework** to calculate a deployment policy π^*_I that is robust against the worst- case adaptive adversary.
- Mathematical Impact: CVPAO* constructs an optimal defense that is secure even if the attacker has complete knowledge of the defender's system topology, neutralizing the adversary's capacity to bypass critical bottlenecks:

$$\pi^*_I = \arg\min_{\pi_I} \max_{\pi_A} \min_{v \in V} [C(v; \varphi, \Sigma) - \delta(v)] \tag{57}$$

✓ *Event-Driven Feedback Adaptability*

Security topologies are highly dynamic; new patches, active host compromises, or changed asset values modify the graph continuously.

- The Achievement: The integration of a real-time reactive feedback loop (Steps 10–12) monitors incoming authenticated updates, recomputing paths instantly upon verified events. This guarantees that defensive configurations adapt on the fly without manual intervention.

- *Comparative Performance Matrix*

Table 5 highlights the theoretical and operational advantages of CVPAO* relative to legacy search and planning frameworks.

Table 5 Comparative Profile of Algorithmic Paradigms

Metric / Feature	**Classic Dijkstra** / A_*	**Traditional** AO_*	**CVPAO* (Proposed)**
Graph Paradigm	Simple OR trees	Directed AND-OR graphs	AND-OR Attack Hypergraphs
Trust Paradigm Temporal Modeling Threat Interaction	Explicit trust (No validation) Static edge cost values Static environment assumption	Explicit trust (No validation) Static edge cost values Static environment assumption	Zero-Trust (ECDSA Validated) Dynamic Attacker vs. Interceptor Delay Minimax Active Game Formulation
Response Quality	Reactive routing	Static path isolation	Dynamic Proactive Interception
Real-Time Loops	Static execution	Static execution	Event-Driven Adaptive Rescheduling

➢ *Final Conclusions*

- *Unified Mathematical Framework Achieved*

This thesis successfully formulates the interception problem as a unified spatio-logical optimization system:

$$S = (G, \varphi, \Sigma, A, I)$$

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

Where logical structure, spatial dynamics, and cryptographic validity are integrated into a single model.

✓ *Conclusion*

Security is no longer treated as separate components (graph, motion, cryptosystem), but as a single mathematically consistent system.

- *Role of φ : Logical Structure is Fundamental*

The introduction of the node-type function:

$$\varphi : V \rightarrow \{AND, OR, LEAF, GOAL\}$$

Enables precise modelling of real-world attack dependencies.

✓ *Conclusion*

- AND nodes enforce multi-condition breaches
- OR nodes model alternative attack strategies

Thus, φ transforms the problem from simple pathfinding into logical reasoning over constraints.

- *AO* Computes True Attacker Cost*

Using φ, the AO* cost recursion:

$$C(v) = \begin{cases} \min, & OR \\ \Sigma, & AND \end{cases}$$

Was proven (Lemma) to satisfy:

$$C(v) = \tau(v)$$

✓ *Conclusion*

AO* does not merely approximate—it computes the actual minimum attack time under logical constraints. This is a critical theoretical result.

- *Predictive Interception is Provably Superior*

The main theorem establishes:

$$\exists v^* \text{ such that } \delta(v^*) \leq \tau(v^*)$$

And hence:

$$TTIAO_* < TTIReactive$$

✓ *Conclusion*

Unlike reactive systems, which respond after intrusion progression, the proposed system:

- predicts attacker trajectory
- identifies mandatory logical bottlenecks
- enables early interception

- *Cryptographic Integrity Ensures Trustworthy Decisions By incorporating ECDSA:*

$$E^* = \{(v, u) \in E : Verify(Sign) = 1\}$$

Only authenticated transitions are considered.

✓ *Conclusion*

The system operates under a zero-trust assumption, ensuring:

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

- resistance to data tampering
- protection against false state injection
- reliability of computed interception strategies

- *Optimal Interception Formulated as Minimax Problem*
  The final formulation:

$$\min_{\pi_I} \max_{\pi_A} \min_{v \in V} [C(v; \varphi, \Sigma) - \delta(v)]$$

✓ *Conclusion*
The interception problem is rigorously reduced to a minimax optimization, capturing:

- worst-case attacker behaviour
- optimal defender strategy

- *Practical Impact: Reduction in Time-To-Interception By combining:*

✓ AO* (predictive reasoning)
✓ φ (logical constraints)
✓ Σ (cryptographic validation) The system achieves:
✓ earlier interception points
✓ reduced computational overhead (via improved ECDSA in your PDF)
✓ improved real-time response

✓ *Conclusion*
The framework provides a provable reduction in interception latency while maintaining computational efficiency.

➢ *Conceptual Contribution*
This thesis introduces a key paradigm shift:

Security is not just monitoring movement—it is predicting logical inevitability under constrained dynamics.

➢ *Limitations*

- Deterministic attacker model (no stochastic behaviour)
- Static graph assumptions between updates
- Heuristic design impacts AO* efficiency

➢ *Future Work*

- Stochastic AO* (probabilistic attacker paths)
- Game-theoretic equilibrium models
- Continuous-space control
- Real-time distributed implementation
- Development of time complexity and space complexity of the final optimization algorithm

➢ *Research Contribution*

The principal contribution of this thesis is the development of an integrated mathematical framework for minimizing interception latency in high-security facilities through intelligent decision-making. Unlike conventional surveillance systems that rely on static patrol routes and reactive responses, the proposed framework models the security environment as an AND–OR graph and applies the AO* optimization algorithm to predict an intruder's most probable attack path and identify optimal interception points.

To ensure the integrity of security-state transitions, the framework incorporates cryptographic authentication using digital signatures, thereby preventing malicious manipulation of sensor-generated information. The proposed architecture further integrates graph-based anomaly detection, secure communication, and autonomous interception into a unified decision-support framework capable of operating in real time.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

A significant contribution of this research is the formulation of a mathematical optimization model that jointly minimizes interception cost, response delay, communication latency, and resource utilization. Based on this formulation, a final optimization algorithm is developed and theoretically validated through mathematical analysis, including correctness arguments and computational complexity evaluation. The proposed framework therefore provides a scalable, secure, and mathematically justified approach for AI-enabled protection of high-value assets while substantially reducing interception latency in autonomous surveillance environments.

The proposed framework extends beyond the direct application of existing optimization and cryptographic techniques by establishing a rigorous mathematical foundation for autonomous interception planning. The research derives a formal cost function, constructs an admissible heuristic for AO* search, and proves the validity of graph-state updates under dynamic threat conditions. Furthermore, the proposed optimization algorithm guarantees systematic decision-making by balancing security constraints, communication overhead, and interception efficiency. This integration of graph theory, heuristic optimization, cryptographic verification, and mathematical analysis provides a comprehensive and practical framework for nextgeneration intelligent surveillance systems capable of securing critical infrastructure against both physical and cyber-enabled threats.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

REFERENCES

[1]. N. Koblitz, Algebraic Aspects of Cryptography, Algorithms and Computation in Mathematics, Vol. 3, Springer, Berlin, Heidelberg, 1998.

[2]. M. Lavanya and V. Natarajan, "LWDSA: Light-weight Digital Signature Algorithm for Wireless Sensor Networks," S¯adhan¯a, vol. 42, no. 10, pp. 1629–1643, 2017.

[3]. X. Cui and H. Shi, "A*-based Pathfinding in Modern Computer Games," International Journal of Computer Science and Network Security, vol. 11, no. 1, pp. 125–130, Jan. 2011.

[4]. S. Gong, "Recent Progress in ECDSA and DSA Digital Signature Algorithms," Transactions on Computer Science and Intelligent Systems Research, vol. 6, Proceedings of ISET 2024, Warwick Evans Publishing, 2024.

[5]. D. Davidov and S. Markovitch, "Multiple-Goal Heuristic Search," Journal of Artificial Intelligence Research, vol. 26, pp. 417–451, 2006.

[6]. L. Akoglu, H. Tong, and D. Koutra, "Graph Based Anomaly Detection and Description: A Survey," Data Mining and Knowledge Discovery, vol. 29, no. 3, pp. 626–688, Springer, 2015.

[7]. D. Sensarma, "Graph Based Anomaly Detection Techniques: A Review," in Science and Technology – Recent Updates and Future Prospects, vol. 7, Chapter 4, Book Publisher International, 2024.

[8]. C. C. Noble and D. J. Cook, "Graph-Based Anomaly Detection," in Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '03), pp. 631–636, ACM, 2003.

[9]. A. Martelli and U. Montanari, "Additive AND/OR Graphs," in Proceedings of the 3rd International Joint Conference on Artificial Intelligence (IJCAI), Stanford, California, USA, Aug. 1973, pp. 1–11.

[10]. X. Hu, M. Yang, C. Liu, W. Cai, et al., "Intelligent Unmanned Defense System for Autonomous Interception of UAVs Based on Improved Acoustic Source Localization Algorithm," IEEE Access, vol. 13, pp. 1–15, Jan. 2025. doi: 10.1109/ACCESS.2025.3575959.

[11]. A. Martelli and U. Montanari, "Optimizing Decision Trees Through Heuristically Guided Search," Communications of the ACM, vol. 21, no. 12, pp. 1025– 1039, Dec. 1978.

[12]. L. Bassham, D. Johnson, and T. Polk, "Representation of Elliptic Curve Digital Signature Algorithm (ECDSA) Keys and Signatures in Internet X.509 Public Key Infrastructure Certificates," Internet Draft, Internet Engineering Task Force (IETF), June 1999.

[13]. T. ElGamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," IEEE Transactions on Information Theory, vol. 31, no. 4, pp. 469–472, July 1985.

[14]. N. Koblitz, "CM-Curves with Good Cryptographic Properties," in Advances in Cryptology – CRYPTO '91, Lecture Notes in Computer Science, vol. 576, Springer-Verlag, 1992, pp. 279–287.

[15]. H. Zhong, R. Zhao, J. Cui, J. Gao, et al., "An Improved ECDSA Scheme for Wireless Sensor Network," International Journal of Future Generation Communication and Networking, vol. 9, no. 2, pp. 73–82, Feb. 2016. doi:10.14257/ijfgcn.2016.9.2.08.

[16]. Y. Gen¸c and E. Afacan, "Design and Implementation of an Efficient Elliptic Curve Digital Signature Algorithm (ECDSA)," in Proceedings of the 2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS), Apr. 2021. doi: 10.1109/IEMTRONICS52119.2021.9422589.

[17]. J. S. Dickens, T. Maweni, T. Setati, and Z. Suddoo, "Design of HERMES: A Mobile Autonomous Surveillance Robot for Security Patrol," MATEC Web of Conferences, vol. 388, Art. 04005, Dec. 2023. doi: 10.1051/matecconf/202338804005.

[18]. N. Ferraz Junior, A. A. A. Silva, A. E. Guelfi, and S. T. Kofuji, "Performance Evaluation of Publish–Subscribe Systems in IoT Using Energy-Efficient and Context-Aware Secure Messages," Journal of Cloud Computing, vol. 11, Art. 6, pp. 1–17, Jan. 2022. doi: 10.1186/s13677-022-00278-6.

[19]. L. Zhang, W. Li, Z. Zhang, Q. Lu, C. Hou, P. Hu, et al., "LogAttn: Unsupervised Log Anomaly Detection with an Autoencoder-Based Attention Mechanism," in International Conference on Knowledge Science, Engineering and Management (KSEM 2021), pp. 222–235, 2021.

[20]. A. Iliev, N. Kyurkchiev, and A. Rahnev, "A Refinement of Knuth's Extended Euclidean Algorithm for Computing Modular Multiplicative Inverse," Communications in Applied Analysis, vol. 25, no. 1, pp. 23–37, May 2021. doi:10.12732/caa.v25i1.3.

[21]. C. P. Schnorr, "Efficient Identification and Signatures for Smart Cards," in G. Brassard (Ed.), Advances in Cryptology – CRYPTO '89, Lecture Notes in Computer Science, vol. 435, Springer-Verlag, 1990, pp. 239–252.

[22]. J. Bresina, R. Dearden, N. Meuleau, S. Ramakrishnan, D. Smith, and R. Washington, "Planning Under Continuous Time and Resource Uncertainty: A Challenge for AI," in Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI 2002), pp. 77–84, 2002.

[23]. E. Benazera, R. Brafman, N. Meuleau, and Mausam, "An AO* Algorithm for Planning with Continuous Resources," in Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05), 2005. [cite: 1, 15]

[24]. A. Konar, Artificial Intelligence and Soft Computing: Behavioral and Cognitive Modeling of the Human Brain, Boca Raton: CRC Press, 1999. [cite: 3, 4]

[25]. F. Ahmad, R. Khan, and A. A. Nkembi, "Improvement in the Interception Vulnerability Level of Encryption Mechanism in GSM," Inventions, vol. 10, no. 4, Art. 56, July 2025. doi: 10.3390/inventions10040056.

[26]. S. Ghosh, "Low-Latency Crypto: An Emerging Paradigm of Lightweight Cryptography," Presentation Proposal, Intel Labs, Intel Corporation, USA.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug221

[27]. T. Padmaja and M. Chandar, "A Comprehensive Survey on Lightweight Cryptography Algorithms to Enhance Quality of Service in Medical Internet of Things," International Journal of Safety and Security Engineering, vol. 15, no. 11, pp. 2403–2417, Nov. 2025. doi: 10.18280/ijsse.151120.
[28]. S. Narmadha and N. V. Balaji, "Improved Network Anomaly Detection System Using Optimized Autoencoder–LSTM," Expert Systems with Applications, vol. 273, Art. 126854, Feb. 2025. doi: 10.1016/j.eswa.2025.126854.
[29]. B. Dowling, M. Fischlin, F. Gu¨nther, and D. Stebila, "A Cryptographic Analysis of the TLS 1.3 Handshake Protocol," Journal of Cryptology, vol. 34, no. 4, Oct. 2021. doi: 10.1007/s00145-021-09384-1.
[30]. A. Shah and M. Siddiqui, "Software Architecture for Machine Learning-Based Systems: Issues, Challenges, and Solutions," Technical Report, Cognitive Lab Camp, June 2023.
[31]. S. Graupner, H. R. Motahari Nezhad, S. Singhal, and S. Basu, "Making Processes from Best Practice Frameworks Actionable," in Proceedings of the Workshops of the 12th IEEE International Enterprise Distributed Object Computing Conference (EDOCW 2009), Auckland, New Zealand, Sept. 1–4, 2009. doi:10.1109/EDOCW.2009.5332021.
[32]. S. Park, J. Y. Lee, and J. Lee, "AI System Architecture Design Methodology Based on IMO (Input–AI Model–Output) Structure for Successful AI Adoption in Organizations," Data & Knowledge Engineering, vol. 150, Art. 102264, Mar. 2024. doi: 10.1016/j.datak.2023.102264.
[33]. A. Bosselaers, "MD4/MD5," in H. C. A. van Tilborg (Ed.), Encyclopedia of Cryptography and Security, Springer, Boston, MA, 2005. doi: 10.1007/0-38723483-7 249.
[34]. S. Rao, "Advanced SHA-1 Algorithm Ensuring Stronger Data Integrity," International Journal of Computer Applications, vol. 130, no. 8, pp. 25–27, Nov. 2015. doi: 10.5120/ijca2015907056.
[35]. S. K. Devineni, S. Kathiriya, and A. Shende, "Machine Learning-Powered Anomaly Detection: Enhancing Data Security and Integrity," Journal of Artificial Intelligence & Cloud Computing, vol. 2, no. 2, pp. 1–9, May 2023. doi:10.47363/JAICC/2023(2)184.
[36]. M. Song, "Cryptographic Innovations for Data Integrity in Cloud and IoT Environments," 2024.
[37]. B. Mohan Sai and M. Bhatia, "A Survey on IoT Security Using Cryptographic Algorithms," E3S Web of Conferences, vol. 453, Art. 01048, Nov. 2023. doi:10.1051/e3sconf/202345301048.
[38]. M. S. Sozol, G. M. Saki, and M. M. Rahman, "Anomaly Detection in Cybersecurity with Graph-Based Approaches," International Journal of Scientific Research in Engineering and Management (IJSREM), vol. 8, no. 8, pp. 1–5, Aug. 2024. doi: 10.55041/IJSREM37061.
[39]. W. Wang, Y. Shang, Y. He, Y. Li, and J. Liu, "BotMark: Automated Botnet Detection with Hybrid Analysis of Flow-Based and Graph-Based Traffic Behaviors," Information Sciences, vol. 511, pp. 284–296, 2020. doi: 10.1016/j.ins.2019.09.024.
[40]. N. Dilini, N. Sun, Y. Miao, and N. Moustafa, "A Comprehensive Review on Graph-Based Anomaly Detection: Approaches for Intrusion Detection," Applied Sciences, vol. 16, no. 4, Art. 1906, 2026. doi: 10.3390/app16041906.
[41]. S. Lamba and M. Sharma, "An Efficient Elliptic Curve Digital Signature Algorithm (ECDSA)," in Proceedings of the 2013 International Conference on Machine Intelligence and Research Advancement (ICMIRA), Katra, India, 2013, pp. 179–183. doi: 10.1109/ICMIRA.2013.41.
[42]. C. Alioanei and N. Popescu, "AI-Based Solutions for Security and Resource Optimization in IoT Environments: A Systematic Review," Information, vol. 16, no. 10, Art. 841, 2025. doi: 10.3390/info16100841.
[43]. A.-M. T. Ehis, "Optimization of Security Information and Event Management (SIEM) Infrastructures, and Events Correlation/Regression Analysis for Optimal Cyber Security Posture," Archives of Advanced Engineering Science, July 2023. doi: 10.47852/bonviewAAES32021068.
[44]. K. Ikpe and E. Ashigwuike, "Enterprise Network Design and Security Optimization," Open Access Library Journal, vol. 12, no. 3, pp. 1–17, Jan. 2025. doi: 10.4236/oalib.1112489.
[45]. A. A. A. Lateef, H. A. El Shenbary, A. A. Gouda, and M. A. Razek, "Optimization Algorithms for Path Routing and Security in Software-Defined Networking (SDN)," International Journal of Computing and Digital Systems, vol. 15, no. 1, pp. 1–15, Jan. 2026. Received Sept. 2, 2025; revised Nov. 22, 2025; accepted Dec. 2, 2025.
[46]. S. Sivakumar, A. Lawrence, and A. L. Selvakumar, "Time Complexity Analysis and Comparison of SHA Algorithms," Journal of Advanced Research in Dynamical and Control Systems, vol. 11, no. 2, pp. 1922–1927, Feb. 2019. doi:10.7717/peerj-cs.981.
[47]. K. Ali, F. Akhtar, S. A. Memon, A. Shakeel, A. Ali, and A. Raheem, "Performance of Cryptographic Algorithms Based on Time Complexity," in Proceedings of an IEEE Conference, IEEE, 2023.
[48]. N. Hasan and P. B. Gari, "Time-Complexity Characterization of NIST Lightweight Cryptography Finalists," arXiv preprint arXiv:2602.05641 [cs.CR], 2026. doi: 10.48550/arXiv.2602.05641.