NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

LS-AODV: A ROUTING PROTOCOL BASED ON LIGHTWEIGHT CRYPTOGRAPHIC TECHNIQUES FOR A FANET OF NANO DRONES

by

Cody W. Vernon

March 2022

Thesis Advisor: Preetha Thulasiraman
Second Reader: Murali Tummala

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** March 2022	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE
LS-AODV: A ROUTING PROTOCOL BASED ON LIGHTWEIGHT CRYPTOGRAPHIC TECHNIQUES FOR A FANET OF NANO DRONES

5. FUNDING NUMBERS
RMQ80

6. AUTHOR(S) Cody W. Vernon

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
ONR

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release. Distribution is unlimited.

12b. DISTRIBUTION CODE
A

13. ABSTRACT (maximum 200 words)

With the battlespace rapidly shifting to the cyber domain, it is vital to have secure, robust routing protocols for unmanned systems. Furthermore, the development of nano drones is gaining traction, providing new covert capabilities for operators at sea or on land. Deploying a flying ad hoc network (FANET) of nano drones on the battlefield comes with specific performance and security issues. This thesis provides a novel approach to address the performance and security concerns faced by FANET routing protocols, and, in our case, is specifically tailored to improve the Ad Hoc On-Demand Distance Vector (AODV) routing protocol. The proposed routing protocol, Lightweight Secure Ad Hoc On-Demand Distance Vector (LS-AODV), uses a lightweight stream cipher, Trivium, to encrypt routing control packets, providing confidentiality. The scheme also uses Chaskey-12-based message authentication codes (MACs) to guarantee the authenticity and integrity of control packets. We use a network simulator, NS-3, to compare LS-AODV against two benchmark routing protocols, AODV and the Optimized Link State Routing (OLSR) protocol, in order to gauge network performance and security benefits. The simulation results indicate that when the FANET is not under attack from black-hole nodes, LS-AODV generally outperforms OLSR but performs slightly worse than AODV. On the other hand, LS-AODV emerges as the protocol of choice when a FANET is subject to a black-hole attack.

14. SUBJECT TERMS
unmanned systems, lightweight stream cipher, black-hole attack, NS-3, flying ad hoc network, FANET, cybersecurity, routing protocol, Lightweight Secure Ad Hoc On-Demand Distance Vector, LS-AODV, message authentication codes, MACs

15. NUMBER OF PAGES
121

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

LS-AODV: A ROUTING PROTOCOL BASED ON LIGHTWEIGHT CRYPTOGRAPHIC TECHNIQUES FOR A FANET OF NANO DRONES

Cody W. Vernon
Lieutenant, United States Navy
BS, United States Naval Academy, 2015

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March 2022

Approved by: Preetha Thulasiraman
Advisor

Murali Tummala
Second Reader

Douglas J. Fouts
Chair, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

With the battlespace rapidly shifting to the cyber domain, it is vital to have secure, robust routing protocols for unmanned systems. Furthermore, the development of nano drones is gaining traction, providing new covert capabilities for operators at sea or on land. Deploying a flying ad hoc network (FANET) of nano drones on the battlefield comes with specific performance and security issues. This thesis provides a novel approach to address the performance and security concerns faced by FANET routing protocols, and, in our case, is specifically tailored to improve the Ad Hoc On-Demand Distance Vector (AODV) routing protocol. The proposed routing protocol, Lightweight Secure Ad Hoc On-Demand Distance Vector (LS-AODV), uses a lightweight stream cipher, Trivium, to encrypt routing control packets, providing confidentiality. The scheme also uses Chaskey-12-based message authentication codes (MACs) to guarantee the authenticity and integrity of control packets. We use a network simulator, NS-3, to compare LS-AODV against two benchmark routing protocols, AODV and the Optimized Link State Routing (OLSR) protocol, in order to gauge network performance and security benefits. The simulation results indicate that when the FANET is not under attack from black-hole nodes, LS-AODV generally outperforms OLSR but performs slightly worse than AODV. On the other hand, LS-AODV emerges as the protocol of choice when a FANET is subject to a black-hole attack.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

AES	Advanced Encryption Standard
AODV	Ad Hoc On-Demand Distance Vector
ARX	Add-Rotate-XOR
BATMAN	Better Approach to Mobile Ad Hoc Network
DCF	Data Collection Framework
DFA	Differential Fault Analysis
DOD	Department of Defense
DSR	Dynamic Source Routing
DSDV	Destination-Sequenced Distance Vector Routing
eSTREAM	ECRYPT Stream Cipher Project
FANET	flying ad hoc network
HMAC	Hash-based message authentication codes
ICACTA	International Conference on Advanced Computing Technologies and Applications
IETF	Internet Engineering Task Force
INRIA	French National Institute for Research in Computer Science and Control
IV	initialization vector
LCAD	Load Carry and Deliver Routing
LFSR	Linear-Feedback Shift Register
LS-AODV	Lightweight Secured Ad Hoc On-Demand Distance Vector

MACs	message authentication codes
MANET	mobile ad hoc network
MAODV	Secure Modified Ad Hoc On-Demand Distance Vector
m/s	meter per second
ms	millisecond
MPRs	multipoint relays
NPS	Naval Postgraduate School
NRL	U.S. Naval Research Laboratory
NSA	National Security Agency
NS-3	Network Simulator 3
OLSR	Optimized Link State Routing
ONR	Office of Naval Research
PDR	packet delivery ratio
QoS	Quality of service
RERR	Route Error
RFC	Request for Comments
RREP	Route Reply
RREQ	Route Request
SHARP	Sharp Hybrid Adaptive Routing Protocol
TC	Topology Control
TORA	Temporally Ordered Routing Algorithm
TS-AODV	Time-Slotted Ad Hoc On-Demand Distance Vector

UAVs unmanned aerial vehicles

VANET vehicular ad hoc network

ZRP Zone Routing Protocol

THIS PAGE INTENTIONALLY LEFT BLANK

Acknowledgments

First and foremost, I must thank my wife, Halley, for her endless support throughout my studies. While the COVID-19 pandemic presented new challenges during our time at Naval Postgraduate School, she took care of our family. She frequently pushed me to overcome any hurdles I faced while working on this thesis.

I would also like to thank Dr. Preetha Thulasiraman for her continued support and constant enthusiasm. I cannot express my gratitude enough for her expert guidance and commitment throughout this endeavor. I'm genuinely proud of this thesis, and it would not have been possible without her guiding hand.

Thank you to Dr. Murali Tummala and Michele D'Ambrosio for your persistent patience and examination of this paper.

Finally, a special thanks to Alison Scharmota at the Graduate Writing Center for helping career naval officers develop essential writing skills.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

A flying ad hoc network (FANET) consists of flying nodes, such as unmanned aerial vehicles (UAVs) that communicate and cooperatively accomplish complex tasks without wired connections. These flying nodes require a decentralized communication architecture to operate under fast-paced, dynamic conditions. A FANET has several distinguishing features compared to a mobile ad hoc network (MANET) or vehicular ad hoc network (VANET). The differences between FANETs and similar ad hoc networks includes:

- Three-dimensional spatial movement (compared to VANETS that primarily operate on fixed roads),
- Higher node density (compared to VANETS operating in an urban or rural setting),
- Higher speed ranges of 30 to 460 km/h (compared to driving speed ranges of 10 to 120 km/h),
- Lower energy storage capacities. [1]

1.1 Application of Flying Ad Hoc Networks

The deployment of FANETs is gaining traction across many military and civilian domains. Because wired infrastructure is not required, FANETs can be invaluable to personnel on the ground due to their ability to be deployed rapidly from outside the combat zone. Therefore, these specialized networks are exceedingly suited for disaster response, search and rescue operations, and military missions.

1.1.1 Military Applications

In 2009, the Department of Defense (DOD) stated that UAVs flew over 450,000 hours in support of critical missions, including Operation Enduring Freedom and Iraqi Freedom [2]. UAVs are particularly useful in theaters of conflict where human lives are endangered. A multi-UAV system can provide real-time monitoring and signal analysis over large combat zones, relaying that information to operators on the ground. However, many issues, particularly in the realm of secure communications, need to be resolved before a multi-UAV system

can fully execute its mission while deployed against enemies that possess wireless denial capabilities. At a conference on unmanned vehicle systems in 2020, VADM Jim Kilby, deputy chief of Naval Operations for Warfighting Requirements and Capabilities (OPNAV N9), presented the fundamental problems that the Navy faces in deploying a multi-UAV system. Kilby noted:

> The ability to network and control, C2, all those unmanned vehicles is significant and important. Think about the aggregation of that demand on the network and understanding that making sure we field a network that's robust enough to handle all our vehicles in many different types of environments [3].

A modern military requires reliable and secure communication networks that allow force commanders to conduct a multitude of warfare operations, such as surveillance or weapons deployment.

1.1.2 Civilian Applications

As the information age progresses, more and more civilian applications become dependent on network connectivity. FANETs provide an alternative method to regain network connectivity during natural disasters. Emergency mobile units engaged in reconnaissance or search and rescue can significantly increase their coverage by utilizing a FANET architecture. A typical VANET may be restricted during natural disasters due to limited accessibility via roads and waterways while a FANET can operate at high altitudes without mobility impediment [1].

Other roles of FANETs in the civilian sector include crop monitoring, environmental monitoring (pollution levels, wind, humidity, temperature), and police surveillance. They also provide support for other ad hoc networks [1]. For instance, a recent study in the *Journal of Intelligent and Robotic Systems* [4] demonstrated a FANET supervising and regulating traffic in VANETs.

1.2 Research Motivation

The Marine Corps currently deploys the PD-100 Black Hornet Nano Drone, a 1.16-ounce micro-UAV capable of conducting reconnaissance and surveillance operations [5], [6]. This thesis envisions a swarm network of these nano drones deployed over an urban combat zone, cooperatively processing and relaying real-time information to troops on the ground. Nano drones within a military FANET must mitigate the effects of frequent link disconnection, constant network topology changes, and energy storage limitations, all while operating in areas with enemies that possess wireless denial capabilities.

A lightweight yet secure version of the Ad Hoc On-Demand Distance Vector (AODV) routing protocol [7] is a prime candidate to overcome the routing and security challenges faced by a FANET of military-grade nano drones. AODV is one of the most widely used reactive protocols in ad hoc networking. It provides communication between highly mobile nodes with minimal overhead and minimal route establishment latency [8]. Limiting the cryptographic security footprint of a FANET routing protocol allows for critical energy to go toward payload, endurance, and enhanced operational capabilities. Developing an efficient security mechanism also allows a multi-UAV system to react quickly to incoming threats, military or civilian, by maximizing throughput, ensuring a consistent Packet Delivery Ratio (PDR), and minimizing the effect of network jitter and delay.

1.3 Research Contributions

This thesis contributes to funded research by the Office of Naval Research (ONR) to enhance cyber security analytics for cyber-physical systems. This thesis provides a novel approach to address the security concerns faced by FANET routing protocols and is specifically tailored to improve the AODV routing protocol. The proposed scheme uses the lightweight stream cipher, Trivium [9], to encrypt routing control packets, thus providing confidentiality. Second, the scheme uses Chaskey-12 based message authentication codes (MACs) to guarantee the authenticity and integrity of control packets.

The objective of this thesis is to design and assess the performance of the novel Lightweight Secured Ad Hoc On-Demand Distance Vector routing protocol (LS-AODV) in a FANET architecture. This thesis provides the first published attempt at integrating the Trivium stream cipher and Chaskey-12 MACs into a network routing protocol. LS-AODV is com-

pared in terms of various performance metrics to several popular ad hoc routing protocols, specifically Optimized Link State Routing (OLSR) and AODV. The work in this thesis is foundational and serves to reinvigorate the discussion throughout the DOD on the application of lightweight stream ciphers in cyber network systems. Achieving the following four primary objectives constitutes the contributions of this thesis:

- Design a routing protocol security mechanism, using Trivium, a lightweight synchronous stream cipher, and Chaskey-12, a lightweight permutation-based MAC algorithm.
- Integrate the new Trivium and Chaskey-12 based security mechanism into the AODV routing protocol
- Evaluate throughput, delay, jitter, packet loss, and routing overhead performance under optimal network conditions with and without lightweight security.
- Evaluate throughput, delay, jitter, packet loss, and routing overhead performance under black-hole attack conditions and determine optimal network configurations to mitigate adversary denial capabilities.

1.4 Thesis Organization

The remainder of this thesis is organized as follows: Chapter II provides an overview of the network simulator, NS-3, comparative FANET routing protocols, and performance and security concerns that must be addressed. Chapter III discusses the proposed approach to tailor the AODV routing protocol to a military FANET scenario by using the lightweight synchronous stream-cipher Trivium and Chaskey-12 based MACs. Chapter IV describes the simulation setup and parameter selection. Chapter V presents and analyzes the simulation results. Chapter VI concludes the study and recommends options for future work.

CHAPTER 2:
Background and Related Work

This chapter discusses the software used to perform our simulations and provides an overview of FANET routing protocols. First, we discuss the operations and limitations of NS-3, the simulator that models our network. Next, we consider the design considerations and techniques unique to a FANET. Finally, we provide an overview of OLSR, AODV, security concerns specific to AODV, and notable secure versions of AODV.

2.1 NS-3

Network Simulator 3 (NS-3) is a discrete-event driven network simulator that allows researchers to model and analyze software-based network architectures [10]. NS-3 is written primarily in the C++ programming language with optional Python bindings. To write a simulation script, a user must link together various software libraries and build the different layers of a network model. A user defines the individual wireless mobile nodes, the desired mobility and propagation models, the amount of traffic to transmit, and which routing protocol to use. Figure 2.1 depicts the software organization of NS-3.25. This thesis changes the network and protocol software layers to implement a modified version of the AODV routing protocol. Modifications of the helper and test software layers ensure the FANET meets military specifications and that useful network performance metrics for the DOD community are available.

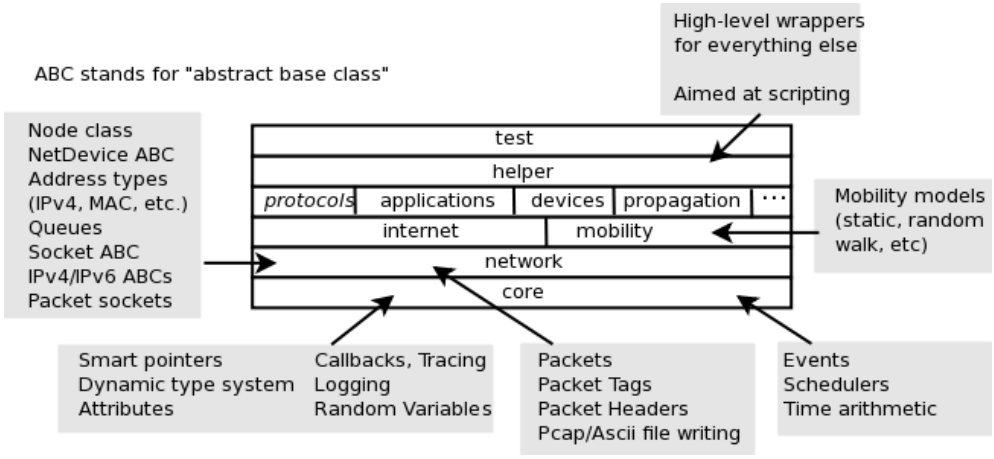

Figure 2.1. NS-3.25 Software Organization. Source: [11].

Figure 2.2 depicts the NS-3 Data Collection Framework (DCF), which allows users to obtain data generated by simulation models, perform post-simulation data processing, and configure various output formats. Output files such as flow monitor files require parsing to extract network performance metrics. The results are available for viewing in the command line terminal or output to an Excel file for compilation. External applications, such as PyViz or NetAnim, allow users to view graphical representations of a network model.

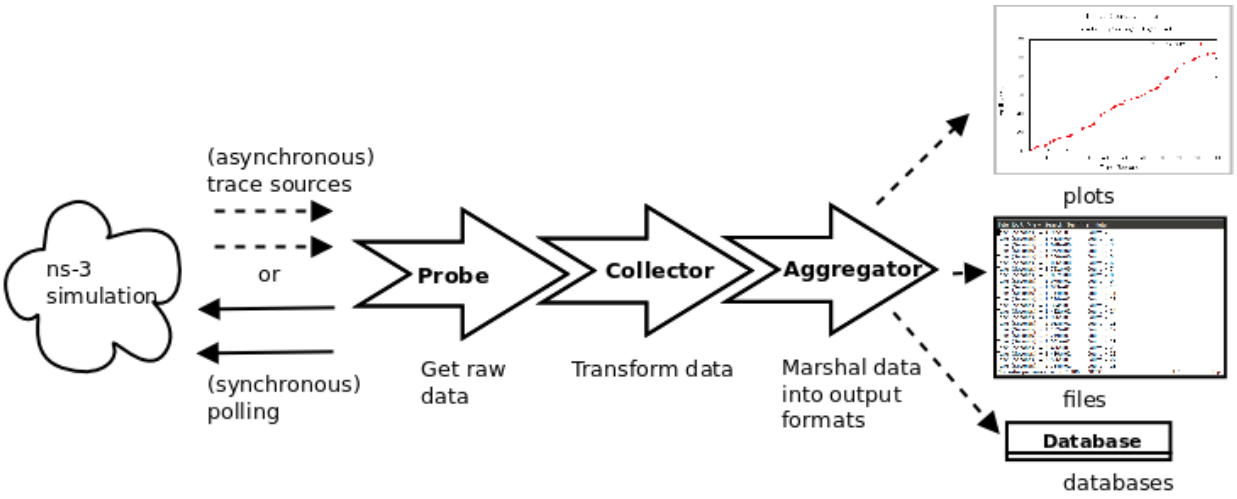

Figure 2.2. NS-3.25 Data Collection Framework. Source: [11].

2.1.1 NS-3 Modeling Limitations

Prior to 2008, a widespread network simulator used for research purposes was NS-2, which focused on reducing compilation time. In 2008, NS-3 was released focusing on scalability and performance rather than compilation time. This shift in simulator philosophy allows for more accurate modeling of network conditions and customization of network tools and scenarios. However, as with any network simulator, modeling may not reflect all real-world conditions. For instance, this thesis uses the NS-3 random waypoint mobility model, which does not allow for node position changes along the Z-axis. Additionally, while the mobility model and propagation models used in this thesis have proved accurate under stringent regression testing, the specific parameters modified to fit a naval use case have not undergone the same level of testing.

2.2 Flying Ad Hoc Network Routing Protocols

Routing protocols govern the connection maintenance between devices in a network. Communication is one of the most challenging issues for multi-UAV systems.

2.2.1 FANET Routing Design Considerations

In the context of FANETs, routing protocols must contend with the adverse effects of high node mobility, limited flight energy resources, and the increasing complexity of UAV mission applications. These unique characteristics of FANETs demand new design considerations.

Mobility

Simulation results and real-world performance of a FANET are strongly correlated with the chosen mobility model. Furthermore, since FANET nodes typically operate at high speeds, a well-chosen mobility model reduces the chance of frequent link disconnections. The node movements within any mobility model are calculated based on solving mathematical equations or reviewing simulation response variables; the latter provides more realistic modeling but with a computational cost. For a UAV swarm conducting military patrol operations, where flexible trajectories are required, the random waypoint mobility model can offer efficient communication [12]. For a UAV swarm conducting autonomous military operations, the Manhattan grid mobility model, which considers geographic restrictions, offers real-time performance improvements [12].

Latency

Whether a FANET is deployed for military or civilian operations, the information communicated between nodes must be transmitted at high rates. Minimizing latency is not only critical to operators on the ground, but also is important to ensure individual UAVs can actively perform collision avoidance. Recent studies indicate that MANET routing protocols may not meet the latency requirements of a FANET with delay-sensitive UAV applications [13].

UAV Platform Constraints

The Marine Corps currently deploys the PD-100 Black Hornet Nano Drone, a 1.16-ounce micro-UAV capable of conducting reconnaissance and surveillance operations [5]. Any FANET routing protocol operating on these nano drones must be energy efficient to combat the lack of energy storage capacity. Space limitations are another constraint on all military

UAVs, particularly the nano variant. The security mechanism of a FANET routing protocol must work efficiently on microcontrollers.

2.2.2 Overview of Routing Protocols

There are plenty of routing protocols used by wireless networks, such as node-centric, data-centric, and pre-computed routing [14]. However, due to the high degree of mobility in a FANET, frequent link disconnection renders these generic ad hoc routing protocols incompatible. Therefore, several classes of routing protocols provide a solution.

- Static routing protocols: All node routing tables are pre-determined prior to flight and not changed during operation.
- Proactive routing protocols: Each node continually updates the routing table with all possible destinations.
- Reactive routing protocol: Each node updates the routing table only when communication with another node is necessary.
- Hybrid routing protocols: All nodes utilize the beneficial features from proactive and reactive protocols [8].

Static routing protocols, such as the Load Carry and Deliver Routing (LCAD) protocol, provide improved communication between a single UAV and a controlling ground-based station when the network topology is fixed. When UAVs have data to transmit to a ground station, they proceed to a pre-determined destination and begin the data transfer process. However, static routing protocols fail to mitigate the problems associated with dynamic network topology changes and UAV-to-UAV communication.

Proactive routing protocols, such as OLSR, Destination-Sequenced Distance Vector Routing (DSDV), and Better Approach to Mobile Ad Hoc Network (BATMAN), allow UAVs to maintain a table with knowledge about all reachable UAVs in the network. The main advantage of proactive routing protocols is that communication between two UAVs experiences minimal delay. However, because each UAV must continually maintain a complete routing table, a large portion of bandwidth is wasted on network maintenance. Furthermore, control packets occupy priority queue space in intermediate nodes, causing systemic congestion and possible data loss [15]. With the disadvantages in mind, proactive routing protocols are typically not used in highly mobile and large FANET applications or when energy

consumption limitations are severe.

On-demand or reactive protocols, such as AODV, Dynamic Source Routing (DSR), and Time-Slotted Ad Hoc On-Demand Distance Vector (TS-AODV), were developed in response to the disadvantages faced by most proactive routing protocols. Reactive protocols use far less bandwidth but suffer from latency issues and possible excessive flooding during the route establishment phase. Due to the highly mobile nature of a multi-UAV swarm, this tradeoff can prove beneficial [16].

2.3 OLSR

The most widely used proactive routing protocol in ad hoc networks is the OLSR protocol. It was developed by the French National Institute for Research in Computer Science and Control (INRIA) and received an experimental Internet Engineering Task Force (IETF) Request for Comments (RFC) in 2003 [17]. The U.S. Naval Research Laboratory (NRL) also maintains an optimized version of OLSR for research and testing purposes [18]. OLSR Version 1 is implemented in NS-3 and has been validated using Wireshark message compliance and unit testing [10].

2.3.1 OLSR Overview

The OLSR routing protocol uses an optimized link-state algorithm to proactively determine the most efficient path between nodes in an ad hoc network. A special characteristic of the OLSR protocol is the use of multipoint relays (MPRs) — neighbor nodes during the flooding process that forward broadcast messages [17]. Each FANET node chooses MPRs to forward their control packets throughout the network. This diffusion technique enhances the flooding process by reducing the transmission count. Furthermore, MPRs periodically announce to the network that they have reachability to the node that selected them. This provides more route opportunities during the path discovery process, allowing for redundancy in highly mobile topology.

The OLSR protocol minimizes network overhead and average network delay by utilizing MPRs. Only partial link state information is flooded to the network during the path discovery process, further reducing the network overhead. Similar to other ad hoc routing protocols,

OLSR flourishes in distributed networks without a central command station. Since each node uses periodic control messages to communicate with destination nodes, the network can recover quickly from lost packet transmission due to collisions or propagation losses. A benefit of the periodicity nature of control message and lack of extra control traffic during link failure events is redundant routes are always available to transmit packets. This redundancy and the use of MPRs makes OLSR particularly suited to handle networks with constant communication between a large number of nodes or when the [source, destination] pairs are changing over time [17].

2.3.2 Neighbor Discovery and Route Calculations

Each node discovers neighboring nodes by periodically broadcasting HELLO messages. These messages are only received and processed by one-hop neighbors. Since each neighbor node has knowledge about its own neighbors, the source node can build tables containing information about its one-hop and two-hop neighbors [17]. The link status between two nodes can be uni-directional, bi-directional, or defined as an MPR.

In order to build a routing table, each node periodically broadcasts Topology Control (TC) messages. These messages contain a list of MPRs selected by the source node. Each node builds a unique topology table from the information contained in TC messages. From this topology table, the optimal routes and backup routes between the source and destination node are determined. In this research, we use OLSR as a comparative routing protocol against the LS-AODV protocol developed in this thesis.

2.4 AODV

One of the most widely used reactive protocols in ad hoc networking is the AODV routing protocol. It was designed by Charles. E Perkins in 1999 and received an experimental IETF RFC in 2003 [7]. One of the goals of the protocol is to minimize the use of network-wide broadcast messages, allowing nodes to respond quickly to link disconnects and network topology changes [7]. With AODV, if a link failure occurs, broadcast messages will be transmitted only if an ongoing communication channel is interrupted. Selective broadcasting significantly reduces the bandwidth usage by control messages, contrary to the operations taken by the OLSR protocol. AODV further reduces network congestion by maintaining a

single route between each source and destination node [7]. The initial concern with this approach is a lack of redundancy; however, as shown in the simulation, the AODV protocol and improved Lightweight Secured Ad Hoc On-Demand Distance Vector (LS-AODV) protocol quickly establish new routes under dynamic network conditions. Since the novel security mechanism in this thesis is designed to improve the AODV routing protocol, this section provides descriptions of the route discovery and maintenance processes, packet formatting, and AODV-specific security concerns.

2.4.1 AODV Route Discovery

Route Request (RREQ) and Route Reply (RREP) are the two message types used in the AODV route discovery process. A node broadcasts a RREQ when it determines that its routing table does not contain a valid route to a destination node. Figure 2.3 depicts the RREQ message format. A RREQ contains the following fields:

- Type: Defines the type of message sent over UDP port 654,
- Hop Count: The number of nodes a packet passes through from source to destination,
- RREQ ID: A unique sequence number relating a RREQ to the originating node,
- Destination IP Address: A unique address identifying the node for which the originating node request a route,
- Destination Sequence Number: The current sequence number indicating a valid route to the destination,
- Originator IP Address: A unique address identifying the node which sent the RREQ,
- Originator Sequence Number: Indicates the freshest route back to the originating node [7].

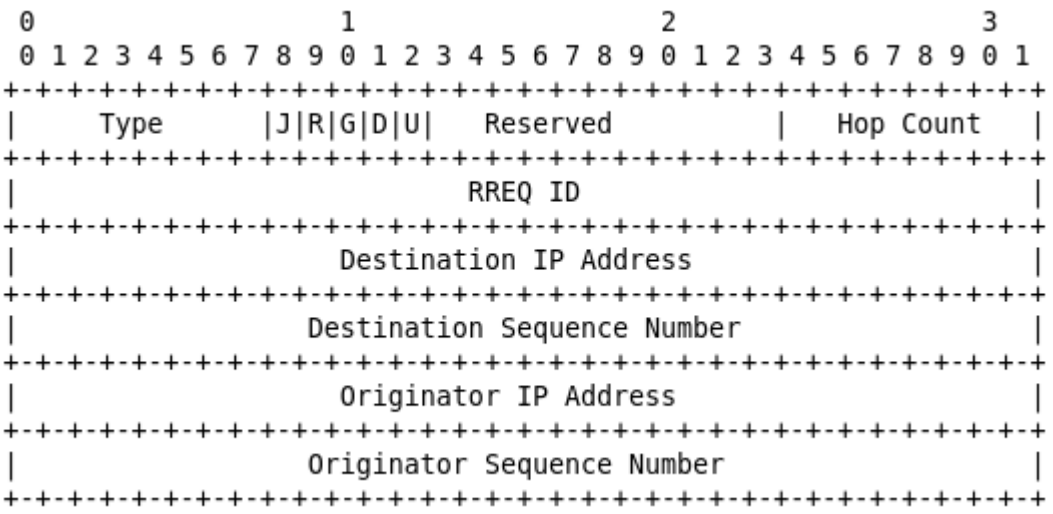

Figure 2.3. Route Request (RREQ) Message Format. Source: [7].

If an intermediate node contains a valid route to the destination, it unicasts a RREP back to the source node [8]. Otherwise, the intermediate node broadcasts the RREQ with an updated hop count to surrounding nodes [8]. When the source node fails to receive a RREP after multiple attempts, it informs the application of the unreachable destination. Figure 2.4 depicts the RREP message format. A RREP contains the following fields:

- Type: Defines the type of message sent over port 654,
- Hop Count: The number of nodes a packet passes through from source to destination,
- Destination IP Address: A unique address identifying the node for which the originating node request a route,
- Destination Sequence Number: A unique identifier for the requested route,
- Originator IP Address: A unique address identifying the node which sent the RREQ,
- Lifetime: A timer which upon expiration directs the node to discard the RREP [7].

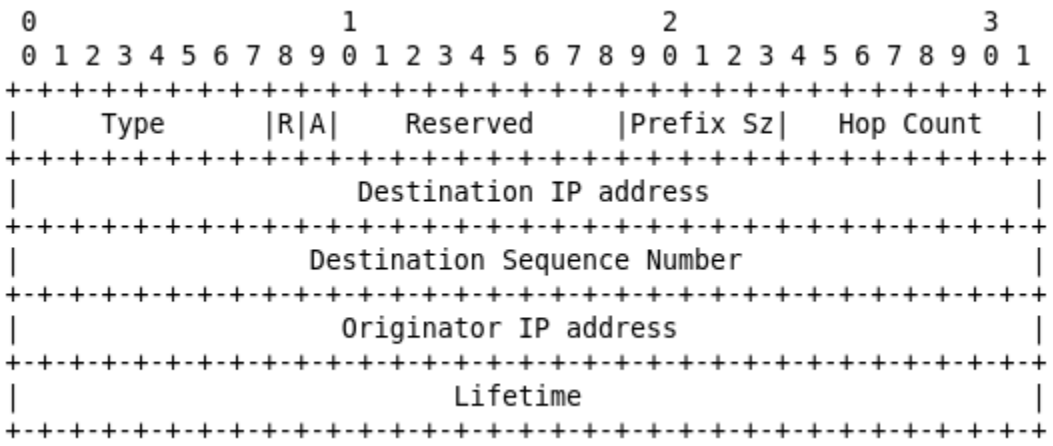

Figure 2.4. Route Reply (RREP) Message Format. Source: [7].

To prevent nodes from flooding the network with RREQs, AODV uses an expanding ring search algorithm [8]. The source node sets the Time to Live (TTL) value in the IP header of the RREQ. If no RREP is received before the TTL expires, the source nose transmits a new RREQ with an incrementally increased TTL value. This process continues until a route is found or a predefined TTL threshold is reached. If a link disconnects and a new route is required, the TTL is initially set to a value corresponding to the hop count [7]. This memory technique further reduces the network overhead. Figure 2.5 illustrates the AODV route discovery process.

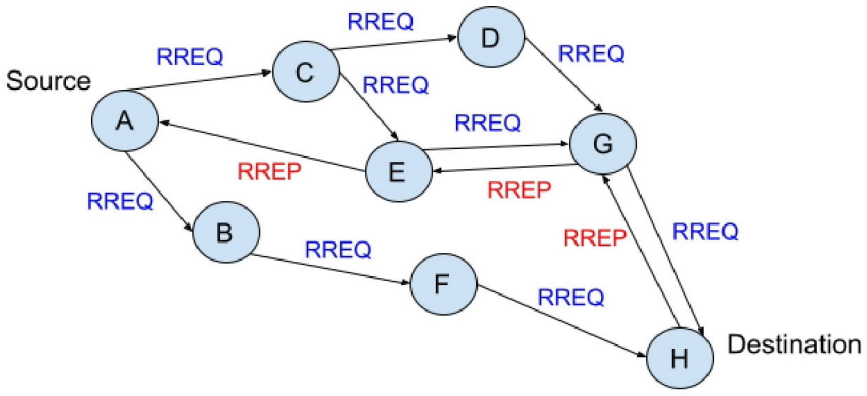

Figure 2.5. AODV Route Discovery Process

2.4.2 AODV Route Maintenance

AODV removes valid routes between nodes only after the source node no longer requires communication with the destination node. The route discovery process restarts if the source node moves out of range of the next link in the active path. The source node receives a Route Error (RERR) message when either the destination node or intermediate node along an active route causes a link disconnection [8]. Figure 2.6 depicts the RERR message. A RERR contains the following fields:

- Type: Defines the type of message sent over port 654,
- DestCount: The number of unreachable nodes listed in the message,
- Unreachable Destination IP Address: A unique address pointing to each unreachable node,
- Unreachable Destination Sequence Number: A unique routing table value corresponding to the destination node [7].

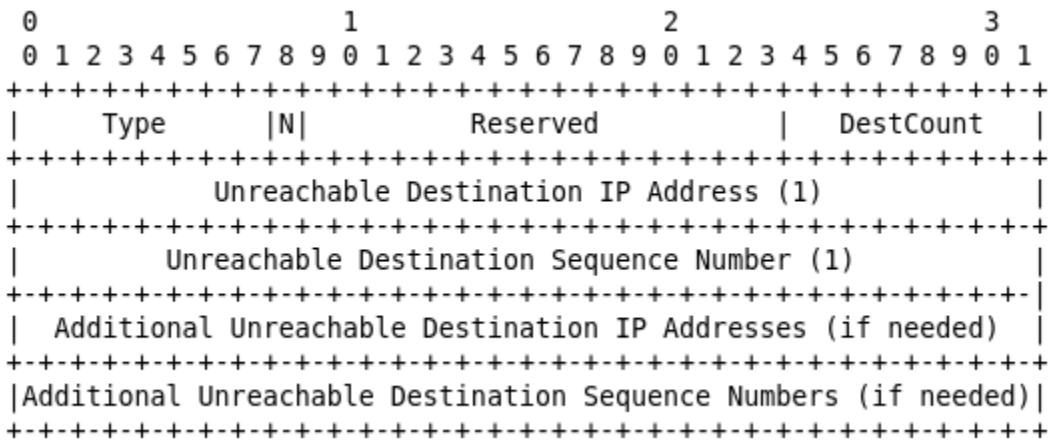

Figure 2.6. Route Error (RERR) Message Format. Source: [7].

The node upstream of the link failure transmits the RERR to neighbors that contain active routes to the destination. When a node receives a RERR, it updates the local routing table with the distance to the destination node set to ∞. When the final RERR reaches the source, it can reinitiate the route discovery process. Entries in the routing that indicate ∞ distance expire after a pre-determined amount of time [8]. This short-term memory technique allows the AODV protocol to continue using the most recent information when making routing decisions.

2.4.3 AODV Security Concerns

The generic AODV protocol and the NS-3 version of AODV do not implement any security measures. This makes FANET networks utilizing the AODV protocol prime targets for black-hole attacks. In a black-hole attack, a malicious node modifies a routing packet sequence number to convince other nodes that it can provide the shortest path to a destination node [19]. When the malicious node receives a communication packet, it fails to forward that packet to the next intermediate node. If an ad hoc network is operating in areas with possible attack vectors, authentication techniques should be implemented to protect control messages [7]. In particular, RREP and RERR messages should be authenticated. If RREP messages are compromised, the creation of incomplete routes between nodes is possible.

If RERR messages are compromised, malicious nodes may disrupt valid routes causing severe network impairment [7]. RFC 3561 also contends that no special considerations, other than adhering to the general protocol specifications, are required to implement a security mechanism for the AODV routing protocol [7]. This contention, factored with AODV performance characteristics, is why this thesis focuses on building a custom security mechanism tailored to improve the AODV routing protocol.

2.5 Secure AODV Routing Protocols

As the AODV routing protocol is one of the most widely used reactive ad hoc protocols, researchers continue to propose new approaches to prevent malicious nodes from comprising AODV based ad hoc networks. This section explores a few proposed security solutions and comments on their applicability to our FANET architecture.

2.5.1 S-AODV

One of the first attempts to incorporate a security mechanism into the AODV routing protocol was conducted at the Nokia Research Center in 2002 [20]. As described in [20], "two mechanisms are used to secure the AODV messages: digital signatures to authenticate the non-mutable fields of the messages, and hash chains to secure the hop count information (the only mutable information in the messages)." Coupled together, these mechanisms are sent alongside AODV control packets as signature extensions. Although S-AODV is robust against a wide range of attacks, including the black-hole attack, it fails to provide hop-by-hop authentication and therefore, an attacker can still increase the hop count hampering each node's routing table decisions.

2.5.2 AODV-MQS

An interesting approach based on blockchain technology was recently proposed in the *EURASIP Journal on Wireless Communications and Networking*. The new routing protocol, AODV-MQS, uses a multipath routing security algorithm to detect and avoid malicious nodes in an ad hoc network [21]. Initial simulation results show AODV-MQS provides an improved packet delivery ratio (PDR) and lower network overhead compared to the original AODV protocol when nodes are operating in an unsafe environment [21]. Blockchain-based

routing protocol development is a relatively new field of study, and more research is needed to determine their applicability to FANETs with limited energy storage reserves.

2.5.3 Modified AODV

A non-cryptographic method to secure the AODV routing protocol from black-hole attacks was proposed at the International Conference on Advanced Computing Technologies and Applications (ICACTA) in 2015 [22]. The proposal modifies the source node operations by creating additional features; a Request Receive Reply function, a new RREP table, and a wait timer. Using expected time metrics, the source node can readily identify malicious nodes and update the routing table to avoid sending data through those nodes [22]. This proposal offers a simple and efficient method to prevent black-hole attacks, but the control packet information remains unencrypted, which does meet the standards of military-based FANETs.

2.5.4 Stream Cipher - AODV

An approach similar to the security mechanism developed in this thesis was presented at the 1st International Conference of Computer and Applied Sciences in 2019 [23]. The proposed scheme uses the Trivium lightweight stream cipher to encrypt control packets while using Hash-based message authentication codes (HMAC) to ensure message authenticity and integrity [23]. Our proposed scheme similarly uses Trivium for encryption but utilizes more efficient message authentication codes based on the Chaskey-12 algorithm.

This chapter discussed the background information necessary to understand the simulation environment used to build a FANET of nano drones. Furthermore, we considered the challenges in designing a FANET routing protocol and explored several routing protocol solutions currently available for implementation.

CHAPTER 3:
Lightweight Secure Ad Hoc On-Demand Distance Vector Routing Protocol

This chapter discusses the proposed scheme used to provide the AODV routing protocol with lightweight security. First, we discuss the design of the Trivium stream cipher and consider its performance and security metrics against the Advanced Encryption Standard (AES) [24]. Next, we discuss the design of Chaskey-12 MACs. Finally, we illustrate how encryption and decryption processes apply to the AODV routing protocol.

3.1 Trivium Encryption

Trivium [9] is a hardware-oriented synchronous stream cipher. Christophe De Cannière and Bart Preneel developed it in 2006 as an attempt to construct the simplest stream cipher without sacrificing security, speed, or flexibility [25]. The ECRYPT Stream Cipher Project (eSTREAM) selected Trivium as one of 3 promising stream ciphers suitable for widespread hardware adoption. In 2012, after no cryptanalytic attacks better than the brute-force attack were shown, the ISO/IEC JTC1 included Trivium in an International Standard for lightweight stream ciphers [26].

Trivium borrows block cipher design techniques to provide reliable security and efficiency. Through the use of a substitution-permutation network, block ciphers prevent linear cryptanalysis by minimizing linear correlations between input and output bits [9]. First, a non-linear substitution box (S-box) exchanges a smaller portion of the original block with another block. Then, linear diffusion layers, more commonly referred to as a permutation box (P-box), permutes each S-box output and feeds the result into the S-box of the next round [27]. Since Trivium acts on one bit at a time, it replaces a block cipher S-box layer with a single S-box. Instead of P-boxes, Trivium maintains non-linear internal states to ensure cipher diffusion and eliminate a separate key addition layer [9].

3.1.1 Trivium Implementation

Trivium uses an 80-bit secret key and an 80-bit initialization vector (IV) to generate up to 2^{64} bits of keystream [25]. The stream cipher process consists of the internal state setup and the keystream generation phase. The following notation will be used to discuss the Trivium implementation and setup.

Notation

$(s_a, ..., s_b)$	internal state of $(a - b + 1)$ bits
s_i	i^{th} shift register bit
z_t	keystream bit generated at time t
$\oplus$	addition over $GF(2)$, i.e. XOR
$\cdot$	multiplication over $GF(2)$, i.e. AND
$\{a, b, c\}$	a^{th} bit, b^{th} bit and c^{th} bit for one round of TRIVIUM.

Internal State Setup

The first phase uses the secret key and IV to initialize the internal state of the stream cipher. A 288-bit initial state is populated with an 80-bit secret key and an 80-bit IV. The remaining 128 bits are set to 0 except for setting the last 3 bits to 1 [25]. This state is rotated over four full cycles equating to 1152 clock cycles. The following pseudo-code describes this process [25] :

$(s_1, s_2, ..., s_{93} \leftarrow (K_1, ..., K_{80}, 0, ..., 0)$
$(s_{94}, s_{95}, ..., s_{177} \leftarrow (IV_1, ..., IV_{80}, 0, ..., 0)$
$(s_{178}, s_{178}, ..., s_{288} \leftarrow (0, ..., 0, 1, 1, 1)$
for i = 1 to 4 x 288 **do**
  $t_1 \leftarrow (s_{66} \oplus s_{91}) \cdot s_{92} \oplus s_{93} \oplus s_{171}$
  $t_2 \leftarrow (s_{162} \oplus s_{175}) \cdot s_{176} \oplus s_{177} \oplus s_{264}$
  $t_3 \leftarrow (s_{243} \oplus s_{286}) \cdot s_{287} \oplus s_{288} \oplus s_{69}$
  $(s_1, s_2, ..., s_{93}) \leftarrow (t_3, s_1, ..., s_{92})$
  $(s_{94}, s_{95}, ..., s_{177}) \leftarrow (t_1, s_{94}, ..., s_{176})$
  $(s_{178}, s_{178}, ..., s_{288}) \leftarrow (t_2, s_{178}, ..., s_{287})$
end for

Key Stream Generation

The second phase consists of extracting 15 specific state bits and utilizing them to iteratively modify three separate state bits and output 1 bit of keystream z_t [25]. The state rotates after each iteration, and this process repeats until the desired amount of keystream bits are output. The following pseudo-code describes this process [25]:

for $i = 1$ to N **do**

$t_1 \leftarrow s_{66} \oplus s_{93}$

$t_2 \leftarrow s_{162} \oplus s_{177}$

$t_3 \leftarrow s_{243} \oplus s_{288}$

$z_i \leftarrow t_1 \oplus t_2 \oplus t_3$

$t_1 \leftarrow t_1 \oplus s_{91} \cdot s_{92} \oplus s_{171}$

$t_2 \leftarrow t_1 \oplus s_{175} \cdot s_{176} \oplus s_{264}$

$t_3 \leftarrow t_1 \oplus s_{286} \cdot s_{287} \oplus s_{69}$

$(s_1, s_2, ..., s_{93}) \leftarrow (t_3, s_1, ..., s_{92})$

$(s_{94}, s_{95}, ..., s_{177}) \leftarrow (t_1, s_{94}, ..., s_{176})$

$(s_{178}, s_{279}, ..., s_{288}) \leftarrow (t_2, s_{178}, ..., s_{287})$

end for

3.1.2 Performance Evaluation

Trivium strives to provide fast and energy-efficient encryption for devices that are power constrained and run applications that require high throughput. Trivium also targets hardware environments with limited gate capacity. In order to provide a compact hardware implementation, Trivium uses a bit-oriented design coupled with the use of non-linear internal states. Fast and power-efficient encryption requires the use of parallelization. The design allows for parallel operation by ensuring that modified state bits only affect subsequent rounds after 66 iterations, therefore, allowing up to 66 iterations to be computed at once [25].

Table 3.1 provides hardware area-based performance metrics for Trivium and AES-128. All designs were tested on ASIC chips with 0.13 μm Standard Cell CMOS libraries. Trivium without parallelization only outperforms AES-128 in terms of total gates required for operation. However, as the amount of parallelization within Trivium increases, the area-

time product and throughput-to-area ratio improve drastically over AES. Table 3.2 indicates that Trivium with four or more parallel iterations outperforms AES-128 in delivering high throughput and bits per cycle. Furthermore, tradeoffs can be made between speed and chip area usage by tweaking the number of parallel operations.

Other studies suggest that RFID tags with Trivium consume approximately eight times less power than RFID tags with AES-128 encryption [28], [29]. The significant speed and area usage results suggest that Trivium can provide a reasonable alternative to AES-128 when used for encryption in a military-grade nano drone.

Table 3.1. Area Based Performance Metrics for Trivium versus AES-128. Adapted from [30, tables 2 and 5].

Design	Area NAND GE, $gates$	Area-Time, μm^2-μs	Tput/Area, $kbps/\mu m^2$
TRIVIUM	2580	133,747	0.0075
TRIVIUM x2	2627	68,092	0.0147
TRIVIUM x4	2705	35,061	0.0285
TRIVIUM x8	2952	19,127	0.0523
TRIVIUM x16	3166	10,259	0.0975
TRIVIUM x32	3787	6,135	0.1630
TRIVIUM x64	4921	3,986	0.2509
AES-128	5398	118,054	0.0085

Table 3.2. Non-Area Based Performance Metrics for Trivium versus AES-128. Adapted from [30, tables 2 and 5].

Design	Throughput, $Mbps$	Bits/Cycle
Trivium	0.100	1
Trivium x2	0.200	2
Trivium x4	0.400	4
Trivium x8	0.800	8
Trivium x16	1.600	16
Trivium x32	3.200	32
Trivium x64	6.400	64
AES-128	0.237	2.37

3.1.3 Security Evaluation

To date, no cryptanalytic attack against Trivium performs better than an exhaustive key search [31]. Table 3.3 provides time metrics and data requirements for the best-known attacks against Trivium. Note that while the security level of Trivium is 2^{80} bits, an exhaustive search requires $\gamma 2^{80}$ bits where γ is the cipher initialization time and is on average 2^{10} bits. Therefore, an exhaustive search requires $\approx 2^{90}$ bits [9].

Trivium reduces two types of linear correlation by implementing several unique features compared to other stream ciphers. Unlike Linear-Feedback Shift Register (LFSR)-based stream ciphers, Trivium ensures the internal state operates nonlinearly, significantly limiting the effectiveness of state recovery attacks that exploit correlations between the keystream bits and internal state bits [9]. Also, the tap positions in Trivium are chosen so that detecting correlations between keystream bits requires at minimum 72 AND gate outputs. These features yield a correlation coefficient of 2^{-72} and force linear distinguisher attacks to take at least 2^{144} bits of keystream, well above the security requirement of 2^{80} bits [9].

A recent study [32] demonstrates a more practical attack that recovers the Trivium secret

key in only six hours. Their approach uses Differential Fault Analysis (DFA) and clock manipulation to attack ASIC implementations of Trivium. However, a DFA attack on a military-grade system is unlikely because an attacker needs direct access to a device with Trivium, access to the keystream output, and can actively restart the device multiple times.

Table 3.3. Cryptanalytical Results for Trivium. Adapted from [30, table 2].

Attack	Time	Data
Linear distinguisher	2^{144}	2^{144}
Guess-and-determine attack	2^{135}	288
Guess-and-determine attack	2^{90}	2^{61}
Solving system of equations	2^{164}	288
Exhaustive key search	2^{80}	80

3.2 Chaskey-12 Message Authentication Codes

MAC algorithms produce a cryptographic checksum that is sent along with an encrypted message to ensure the integrity and authenticity of that message. Chaskey-12 [33] is a lightweight permutation-based MAC algorithm. The permutation structure comes from the SipHash [34] pseudorandom function family. Chaskey-12 is designed for 32-bit microcontroller architectures and employs Add-Rotate-XOR (ARX) block cipher techniques, which means the algorithm is energy and hardware space-efficient. Furthermore, the algorithm is resistant to timing attacks, robust under tag truncation, and does not require a key schedule or the use of nonces [33].

3.2.1 Chaskey-12 Implementation

Chaskey-12 uses a 128-bit secret key to process an arbitrary-length message m and output a MAC of 128-bits or less. The MAC generation process consists of four steps: subkey derivation, message padding, permutation rounds, and optional MAC truncation. The following notation will be used to discuss the Chaskey-12 MAC generation process.

Notation

$+_{32}$	modulo addition 2^{32}
$x <<< s$	rotation of x to the left by s positions
$x << s$	shift of x to the left by s positions
S	nontruncated MAC

Subkey Derivation

From a 128-bit secret key, two 128-bit subkeys K_1 and K_2 are obtained. The following pseudo-code describes this process [35]:

procedure $TimesTwo(a)$
  if $a[127] = 0$ **then**
    return $(a << 1) \oplus 0^{128}$
  else
    return $(a << 1) \oplus 0^{120}10000111$
  end if
end procedure
$K_1 \leftarrow TimesTwo(\mathrm{K})$
$K_2 \leftarrow TimesTwo(\mathrm{K}_1)$

Message Padding

If $|m| \bmod 128 \neq 0$, then a single '1' bit is added to the end of the m followed by 128-d-1 '0' bits, where d is the remainder after dividing the m length by 128. The new padded version of m is denoted by m'.

Permutation Rounds

The padded message m' is divided into 128-bit strings $m_1^{'}, m_2^{'}, ..., m_l^{'}$. Then m' is subjected to a 128-bit permutation π, which consists of 12-round functions. The following pseudo-code describes this process [35]:

procedure $\pi(v)$

$(v_0, v_1, v_2, v_3) \leftarrow v$

for $i = 1$ to r **do**

$v_0 \leftarrow \mathrm{v}_0 +_{32} v_1$

$v_1 \leftarrow \mathrm{v}_1 <<< 5$

$v_1 \leftarrow \mathrm{v}_1 \oplus v_0$

$v_0 \leftarrow \mathrm{v}_0 <<< 16$

$v_2 \leftarrow \mathrm{v}_2 +_{32} v_3$

$v_3 \leftarrow \mathrm{v}_3 <<< 8$

$v_3 \leftarrow \mathrm{v}_3 \oplus v_2$

$v_0 \leftarrow \mathrm{v}_0 +_{32} v_3$

$v_3 \leftarrow \mathrm{v}_3 <<< 13$

$v_3 \leftarrow \mathrm{v}_3 \oplus v_0$

$v_2 \leftarrow \mathrm{v}_2 +_{32} v_1$

$v_1 \leftarrow \mathrm{v}_1 <<< 7$

$v_1 \leftarrow \mathrm{v}_1 \oplus v_2$

$v_2 \leftarrow \mathrm{v}_2 <<< 16$

return $v_0||v_1||v_2||v_3$

end for

$V \leftarrow K$

for $j = 1\ to\ l - 1$ **do**

$V \leftarrow \pi(V \oplus m_i^{'})$

if $nopadding$ **then**

$L \leftarrow K_1$

else

$L \leftarrow K_2$

end if

$S \leftarrow \pi(V \oplus m_l^{'} \oplus L) \oplus L)$

end for

end procedure

Optional Truncation

The t-bit length MAC is derived by taking the t least significant bits of S.

3.2.2 Performance Evaluation

Compared to both hash-function MAC algorithms and block-cipher MAC algorithms, Chaskey-12 produces very little overhead. Hash-function MAC algorithms typically process blocks of at least 512 bits, while Chaskey-12 uses much smaller 128-bit blocks. Operating on smaller blocks allows Chaskey-12 to efficiently move data back and forth between registers and RAM [33]. Block-cipher-based MAC algorithms require key schedule computations, which increases the register pressure. Also, the S-box operations of these algorithms use extensive bit masking operations, further impacting the speed of implementation [33]. Chaskey-12 does not require a key schedule and does not use S-box constructs, avoiding the issues plagued by popular MAC algorithms in use today. Furthermore, by using an ARX permutation structure, hardware implementations of Chaskey-12 require minimal cost and lower design complexity.

In speed-optimized and size-optimized hardware settings, Chaskey-12 computes MACs 8.3 faster than AES-128-CMAC (see Tables 3.4 and 3.5). Also, Chaskey-12 produces MACs 2.1 times faster than the Speck family of lightweight block ciphers developed at the National Security Agency (NSA) [36]. Furthermore, Chaskey-12 uses up to 13 times less energy than Speck block ciphers and 52 times less energy than AES [37].

Table 3.4. Performance Metrics for Chaskey-12 versus AES-128-CMAC (Speed Optimized). Adapted from [33, table 2].

Microcontroller	Algorithm	Data[byte]	ROM size[byte]	Cycles/Byte
Cortex-M0	AES-128-CMAC	128	13492	136.5
Cortex-M0	Chaskey-12	128	1308	21.1
Cortex-M3/M4	AES-128-CMAC	128	28524	105.0
Cortex-M3/M4	Chaskey-12	128	908	8.05

Table 3.5. Performance Metrics for Chaskey-12 versus AES-128-CMAC (Size Optimized). Adapted from [33, table 2].

Microcontroller	Algorithm	Data[byte]	ROM size[byte]	Cycles/Byte
Cortex-M0	AES-128-CMAC	128	11664	140.0
Cortex-M0	Chaskey-12	128	414	19.44
Cortex-M3/M4	AES-128-CMAC	128	10952	89.4
Cortex-M3/M4	Chaskey-12	128	402	12.88

3.2.3 Security Evaluation

The permutation structure of Chaskey-12 can be interpreted as three Even-Mansour [38] block ciphers ($E_{X||Y}(m) = \pi(m \oplus X) \oplus Y$). By making this interpretation, the security of Chaskey-12 reduces down to the 3PRP-indistinguishability of the underlying Even-Mansour block ciphers, which is provably secure [33]. Chaskey-12 remains unbroken and is secure up to $D = 2^{n/2}$ chosen plaintexts and $T = 2^n/D$ queries to the permutation matrix [33]. In practical terms, this means that if the secret key is replaced before processing 2^{64} blocks of 128-bit data, then the security bound will not be reached [35].

While the Even-Mansour security proof thwarts most of the cryptoanalytic attacks Chaskey-12 may face, a few other attacks are possible. For instance, timing attacks are a concern for algorithms running on 32-bit microcontrollers. MAC algorithms, such as UMAC [39], are susceptible to timing attacks because they use data-dependent multiplication instructions. On the other hand, because the execution time of the Chaskey-12 algorithm is constant and does not depend on the secret key K, Chaskey-12 is secure against timing attacks [33]. Since Chaskey-12 uses ARX methodology, it is secure against zero-sum attacks and cube attacks because addition operations ensure each output bit is represented by a high degree polynomial expression. Lastly, while differential paths have been found for a round-reduced version of Chaskey, the full 12-round implementation continues to provide a sufficient security margin against differential cryptanalysis [40].

The original design for Chaskey set the number of permutation rounds to eight. After concerns that eight rounds provide an insufficient security margin, the developers increased the number of permutation rounds to 12. Furthermore, a 16-round version is available that ensures long-term security but at the cost of speed and energy [33].

3.3 Proposed Scheme

The following section describes the integration of the Trivium lightweight synchronous stream cipher algorithm and the Chaskey-12 MAC algorithm into the AODV routing protocol. By using these two mechanisms, the scheme imposes confidentiality, integrity, and authenticity on message traffic across a FANET. The proposed scheme consists of three phases: keystream generation, encryption, and decryption.

3.3.1 Keystream Generation

Trivium and Chaskey-12 are symmetric-key ciphers; the keys of both the sender and receiver must be identical. In the envisioned FANET with nano-drones, the keys are seeded prior to flight. This thesis assumes that the mission duration and overall network data transmission levels are maintained below the key refresh thresholds of Trivium and Chaskey-12.

The Trivium 288-bit internal state is set up using an 80-bit secret key and 80-bit IV. Next, the internal state iteratively updates specific state bits and rotates to produce the required amount of keystream bits. Figure 3.1 depicts the keystream generation process as implemented in this thesis. A more detailed description is available in Section 3.1.

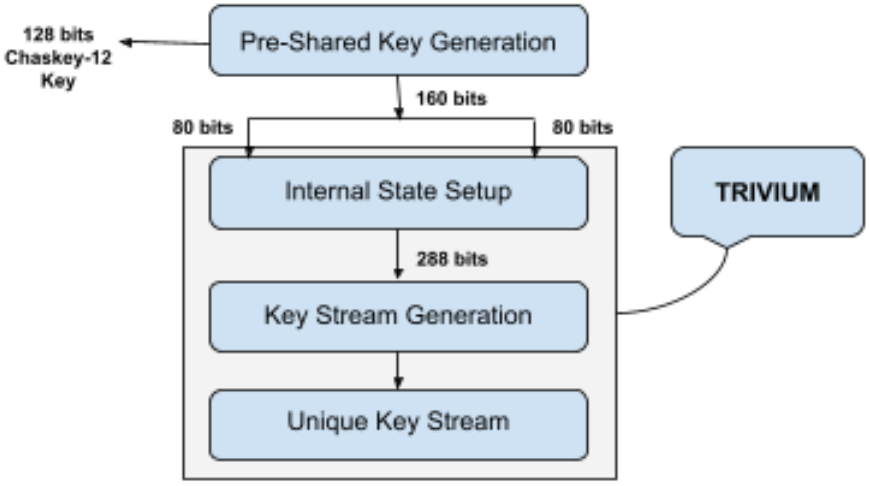

Figure 3.1. Trivium Key Stream Generation Process. Adapted from [23, figure 1].

3.3.2 Encryption Phase

This phase uses the encrypt-then-MAC approach to protect routing control messages. Once the sender node generates a control message, it executes an XOR computation on the message contents with the Trivium keystream and then appends a unique MAC to the end of the message. Figure 3.2 depicts the encryption process. The steps are as follows:

1. $E(M)$ = Trivium keystream $\oplus$ routing control message

2. $MAC(M) = \text{CHASKEY}(K, M)$

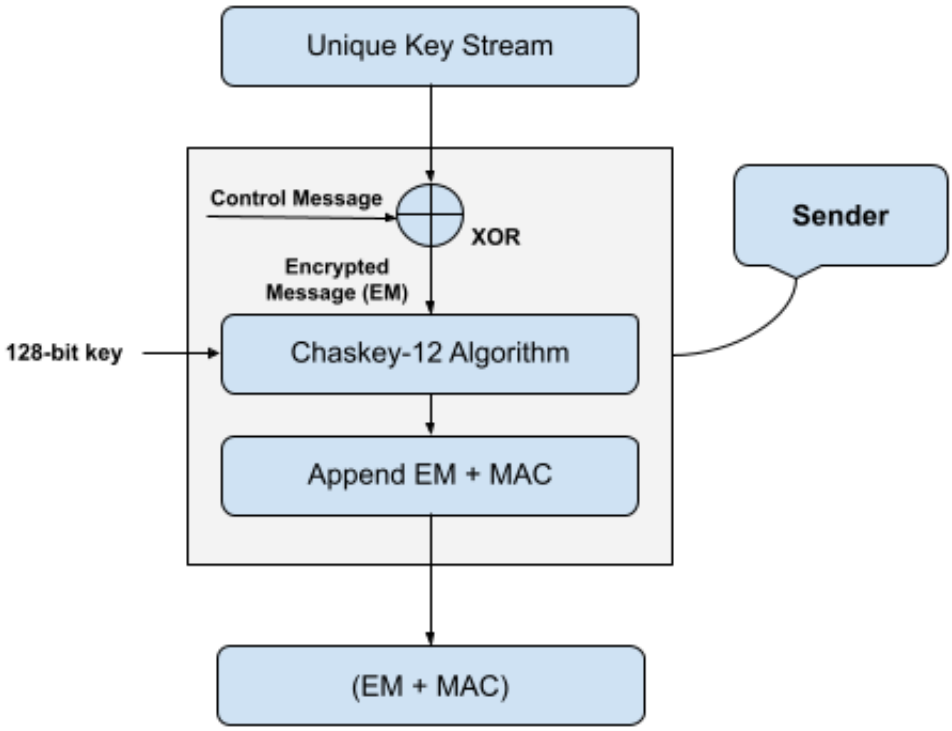

Figure 3.2. LS-AODV Encryption Process. Adapted from [23, figure 2].

3.3.3 Decryption Phase

Control message decryption occurs at each intermediate node and the destination node. Figure 3.3 illustrates the decryption process. When any node receives an encrypted control message, it first strips the 128-bit MAC. Next, the node computes a MAC for the encrypted control message and compares it to the received MAC. If the codes match, then decryption and normal AODV message processing continue; otherwise, the packet is discarded. By verifying MAC agreement prior to decryption, each node saves bandwidth that would be wasted on decrypting faulty messages. Finally, three different actions are possible depending on the node type and the message type.

1. If an intermediate node receives a RREP or RREQ, the node updates routing hop count, re-encrypts new message, calculates new MAC value, and then sends the

encrypted message with the appended MAC to the next node.

2. If an intermediate node receives a RERR message, the node transmits the original encrypted message to the next node.
3. For the destination node, no further encryption or MAC computations are required.

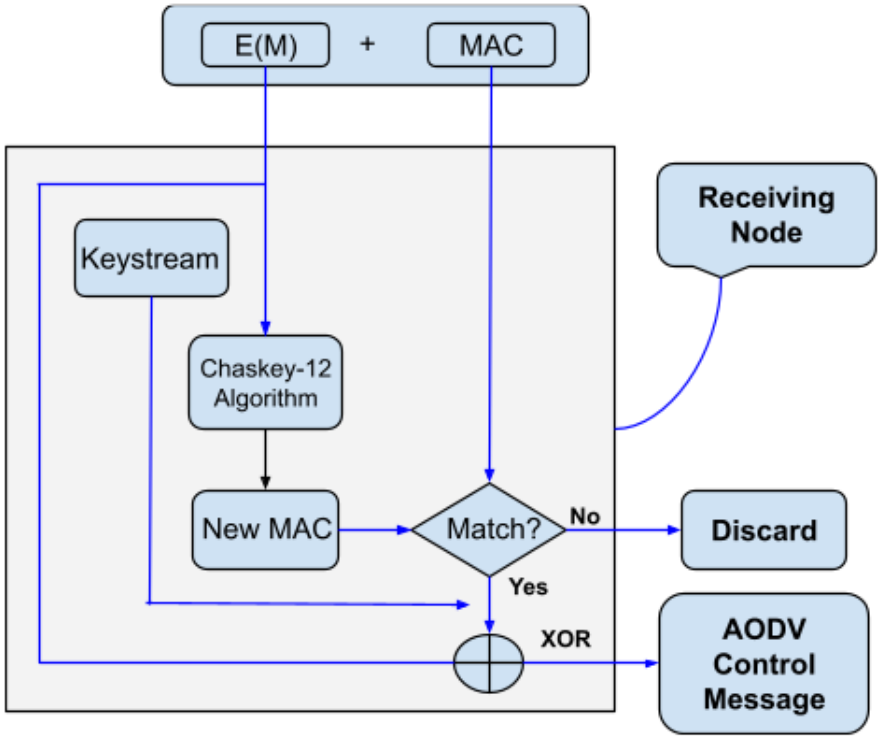

Figure 3.3. LS-AODV Decryption Process. Adapted from [23, figure 3].

This chapter discussed the intricacies of the proposed LS-AODV routing protocol. We detailed the performance and security evaluations of the key components of LS-AODV: the Trivium stream cipher and the Chaskey-12 MAC algorithm. Finally, we described how these cryptographic security footprint techniques are integrated into the AODV routing protocol.

CHAPTER 4:
Simulation Environment and Setup

This chapter discusses the simulation environment chosen to test the LS-AODV routing protocol. First, we highlight which models are available within NS-3 to simulate a FANET. Next, we discuss the performance metrics available to compare OLSR, AODV, and LS-AODV. Lastly, we provide the hardware and software specifications of the system used to perform networking simulations.

4.1 Selection of Network Parameters

When simulating a wireless network, the goal is to mimic the real-world conditions the network will face. NS-3 allows for the customization of a wide range of models. This thesis chose specific mobility, propagation loss, propagation delay, and traffic data rate models. Throughout the simulations, a map size of 670×670 square meters is utilized to simulate nano drones conducting a mission over several city blocks.

4.1.1 Mobility Model

When simulating new ad hoc routing protocols, applying an accurate mobility model is imperative. It is unlikely that a swarm of drones will move at constant speeds and directions in a real-world scenario. The Random Waypoint Mobility model was chosen for this thesis because of its simplicity and ability to accurately model the erratic behavior of drones in a swarm. During the simulation setup, each node is positioned at a random 2-D location in the grid area. When the simulation begins, each node pauses for a specified amount of time. After pausing, each node picks a new random waypoint and random speed and then begins moving towards that destination at the selected speed. When each node reaches its destination, it pauses and recalculates a new speed and waypoint. This process repeats for the duration of the 200-second simulation [10]. This thesis varies the speed and pause time attributes within the Random Waypoint Mobility model to investigate how mobility characteristics affect routing performance.

- Speed - Node speed across grid (5-25 meters/second).

- Pause - Amount of time a node stays at a particular geographical location (2-40 seconds).
- Position Allocator - position model for a destination location of a node (Random-RectanglePositionAllocator). [10]

4.1.2 Propagation Loss and Delay Models

This thesis uses the Two-Ray Ground Propagation Loss model because it provides more realistic loss modeling at longer distances. Separately, for shorter distances, the Friis free-space model is automatically selected [10]. Several attributes are modifiable in the NS-3 Two-Ray Ground Propagation Loss model. This thesis uses the initial recommended values for all attributes.

- Frequency - the propagation carrier frequency (5.15 GHz).
- System Loss - losses experienced by antennas and propagation medium (1).
- Minimum Distance - distance between nodes at which loss is negligible (0.5 meters).
- Height Above Z - antenna height above the node (0 meters). [10]

NS-3 implements two propagation delay models: the Constant Speed Propagation Delay and the Random Propagation Delay models. This thesis chose to implement the Constant Speed Propagation Delay model for several reasons. First, in the random model, all packets experience some random amount of delay. Consequently, the packets may arrive at the receiver out of order and produce another variable to account for in the network analysis. The constant model does not affect the packet order, and while the model assumes a flat Earth, the grid size is small enough that this assumption is acceptable [10]. The only modifiable attribute in the NS-3 Constant Speed Propagation Delay model is the propagation speed.

- Speed - propagation speed through a propagation medium (2.99792×10^8 m/s) [10]

4.1.3 Traffic Data Rate

Rate control algorithms determine the transmission mode and data rate for each packet sent across a wireless network. NS-3 offers 14 real device and theoretical rate control algorithms. Several options include the Constant Rate Wifi Manager, Minstrel Wifi Manager, Rraa Wifi Manager, and the Thompson Sampling Wifi Manager [10]. This thesis uses the Constant

Rate Wifi Manager to ensure that each packet transmits at the same rate. Furthermore, using the constant rate algorithm reduces the impact of optimizations at layers other than the network layer on the simulation results.

4.2 Performance Evaluation Metrics

Quality of service (QoS) is an evaluation of overall network performance. A FANET requires a high degree of QoS due to rapid mobility, frequent link disconnection, and resource constraints. RFC 2501 [41] and RFC 5148 [42] discusses the quantitative metrics available to analyze the QoS of different routing protocols in an ad hoc network environment. This thesis focuses on evaluating throughput, PDR, average network delay, randomly modifying timing (jitter), and routing overhead.

4.2.1 Throughput

Network throughput measures the average amount of bits successfully transmitted from all source nodes to destination nodes. Routing protocols that consistently provide high throughput are desirable because they can efficiently use the allotted bandwidth and limited power reserves. Network throughput is defined as:

$$S = \frac{\sum n_{bi}}{T} \tag{4.1}$$

where n_{bi} is the bits received by each destination node and T is the simulation time.

4.2.2 Packet Delivery Ratio

PDR is the fraction of routing packets received by all destination nodes over the amount of routing packets sent by all source nodes. High PDR indicates that a FANET routing protocol can adapt to a constantly changing network topology and handle dynamic network loading. PDR is defined as:

$$P_{PDR} = \frac{N_{PR}}{N_{PS}} \tag{4.2}$$

where N_{PR} is the total number of packets received and N_{PS} is the total number of packets sent.

4.2.3 Average Network Delay

Average network delay consists of propagation delay, transmission delay, processing delay, and queuing delay. Environmental characteristics affect the propagation delay and each link's data-rate causes transmission delay. Since the simulation uses the same propagation model and data rate for all protocols, the focus is on how different protocols minimize processing and queuing delays. Average network delay is a significant factor in designing a robust and capable routing protocol. FANETs experiencing minimal delay can host low latency applications such as real-time video transmissions. In order to be concise, the term average network delay is written as delay in the follow-on chapters of this thesis. Delay is defined as:

$$D = \frac{\sum(T_{PD} - T_{PS})}{\sum N_{CP}} \tag{4.3}$$

where T_{PD} is the time that packets arrive at the destination node, T_{PS} is the time that packets leave the source node and N_{CP} is the number of connection pairs.

4.2.4 Network Jitter

Network jitter is the average of all delay variations for received packets of the same flow as described in RFC 3393 [43]. Consistently high network jitter can result in packet loss and network congestion. In order to be concise, the term network jitter is written as jitter in the follow-on chapters of this thesis. Jitter is defined as:

$$\sigma_D = \frac{\sum(D_{P_{n-1}} - D_{P_n})}{N_{CP}} \tag{4.4}$$

where $D_{P_{n-1}}$ is the delay amount for the previous packet and D_{P_n} is the delay amount for the current packet.

4.2.5 Routing Overhead

Routing overhead consists of the number of routing control packets sent across the network for route discovery and maintenance purposes. The routing overhead will increase every time a node forwards a packet to either an intermediate node or the destination node. Routing

protocols with low routing overhead typically provide fast route convergence, consume less energy, and allow for scalable network architectures [41].

4.3 Simulation Hardware and Software Specifications

The specifications of the hardware and software used to perform simulations are as follows:

- CPU: 3.4 GHz AMD 4th Gen Ryzen 9 5950X (turbo boost up to 4.9 GHz - 16 cores - 32 threads)
- RAM: 64 GB Dual Channel (DDR4) @ 3200 MHz
- ROM: 1 TB PCIe Gen4 Seq (Read - 7000 MB/s , Write - 5000 MB/s)
- GPU: 10 GB GeForce RTX 3080 w/ 8704 CUDA Cores
- OS: Ubuntu 20.04 LTS (64-bit)
- NS-3: 3.27
- Byte Ordering: Little-Endian

This chapter discussed the models available in NS-3 to simulate a FANET and which network parameters we chose to use throughout the simulation. Also, we described the performance metrics chosen to compare OLSR, AODV, and LS-AODV, namely, throughput, PDR, delay, jitter, and routing overhead. In the next section, we present the simulation results and provide a detailed analysis of the advantages and disadvantages of implementing the LS-AODV routing protocol.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 5:
Simulation Results and Analysis

This chapter discusses the simulation results and analysis for different network configurations. We consider a FANET without malicious nodes and then a FANET with malicious nodes using black hole attack techniques to disrupt network performance. In both cases, we examine the effect of changing mobility, node density, network loading, and pause time on various performance metrics.

5.1 Non-Threat FANET Scenario

In this set of simulations, the goal is to examine the performance of a FANET of nano drones via NS-3. The assumption is that the drones operate in a safe environment without the threat of linked malicious nodes from nefarious actors.

5.1.1 Mobility

This simulation case investigates how changing the node speed affects routing protocol performance. The node speed is adjusted from 5 m/s to 25 m/s in increments of 5 m/s. Twenty nodes move randomly around the grid area, using the Random Mobility model to determine their next position. In order to simulate a highly mobile military environment, each node has a pause time of two seconds. Every node uses the Constant Bit Rate traffic model to send 512-byte data packets with a maximum data rate across the network of 11 Mbps. Each simulation lasts 200 seconds, with the first 100 seconds designated for network setup and route discovery. Table 5.1 summarizes the simulation parameters for the mobility case.

Table 5.1. Simulation Parameters for Mobility Case

Mobility	
Number of Nodes	20
Map Size	670 m × 670 m
Mobility Model	Random Waypoint
Pause Time	2 sec
Speed	5-10-15-20-25 m/s
Simulation Time	200 sec
Traffic	
Traffic Type	Constant Bit Rate
Packet Size	512 bytes
Connection Rate	10 pkts/sec
Routing Protocol	
Protocols	OLSR, AODV, LS-AODV

Figure 5.1 shows the PDR as a function of node mobility. Across the range of mobilities, AODV and LS-AODV generally outperforms OLSR. As the speed of nodes increases, the assumption is that the PDR will drop regardless of the routing protocol chosen. This drop happens because as a node moves faster across a grid, the network topology changes more frequently, resulting in more link failures and therefore reduced PDR. While the LS-AODV routing protocol delivers fewer packets than AODV, it is noteworthy that the PDR for LS-AODV has less variation compared to the other protocols. Therefore, it may excel in military-based FANETs with rapidly changing nano drone speeds, conducting special operations vice patrolling.

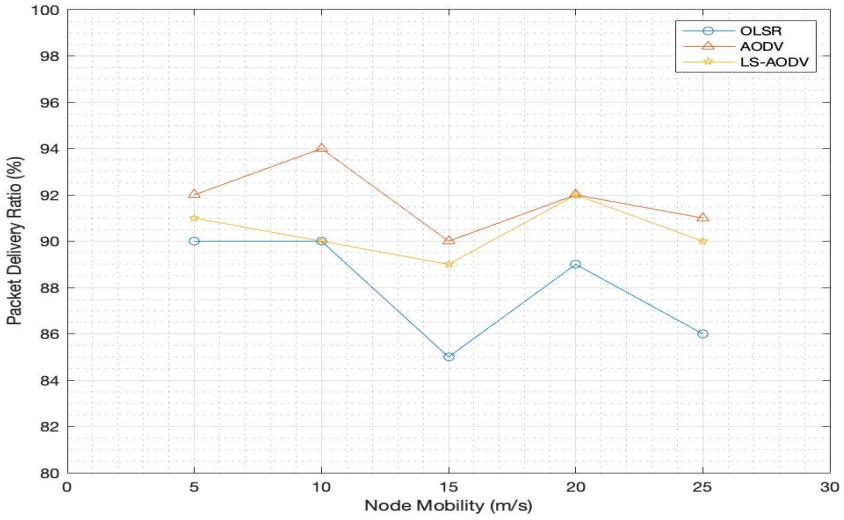

Figure 5.1. Packet Delivery Ratio at Various Speeds. The Node Speed is Adjusted from 5 m/s to 25 m/s in Increments of 5 m/s.

Figures 5.2 and 5.3 depict the network delay and jitter, respectively. Being a proactive routing protocol, OLSR ensures minimal delay between two UAVs communicating in a FANET. OLSR also provides the most consistent delay across various mobility speeds. The packets transmitted using LS-AODV experience the highest delay due to the additional encryption/decryption and MAC generation and verification processes. Without optimizing the transmission process, adding any cryptographic security footprint to an existing routing protocol typically costs higher delay. However, note that LS-AODV only observes a 65.5 millisecond (ms) increase in delay as the node speed increases from 5 to 25 m/s, compared to AODV, which experiences a 220.9 ms increase. At node speeds of 15 m/s and 20 m/s, the gap between LS-AODV and AODV delay is significantly closed. The jitter result indicates similar trends for all three routing protocols. However, at lower node speeds, AODV experiences slightly less jitter than OLSR. The consistently higher jitter incurred by LS-AODV nodes is attributable to longer queue processing times.

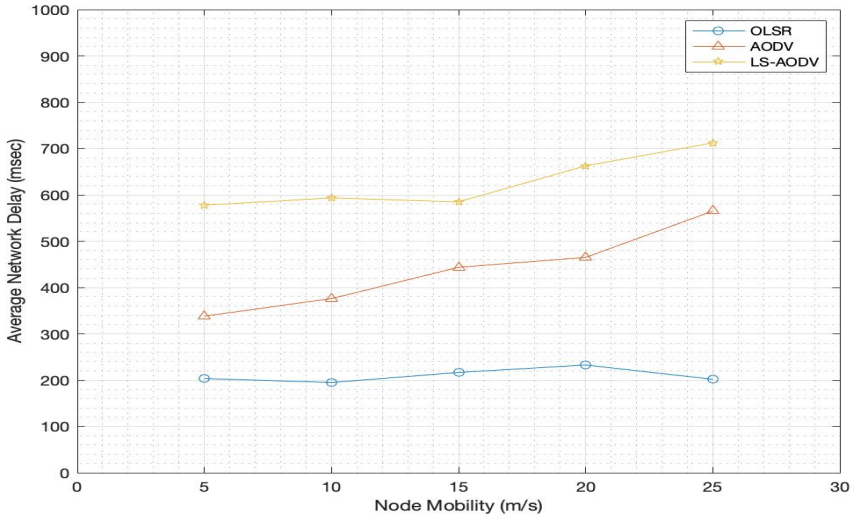

Figure 5.2. Average Network Delay at Various Speeds. The Node Speed is Adjusted from 5 m/s to 25 m/s in Increments of 5 m/s.

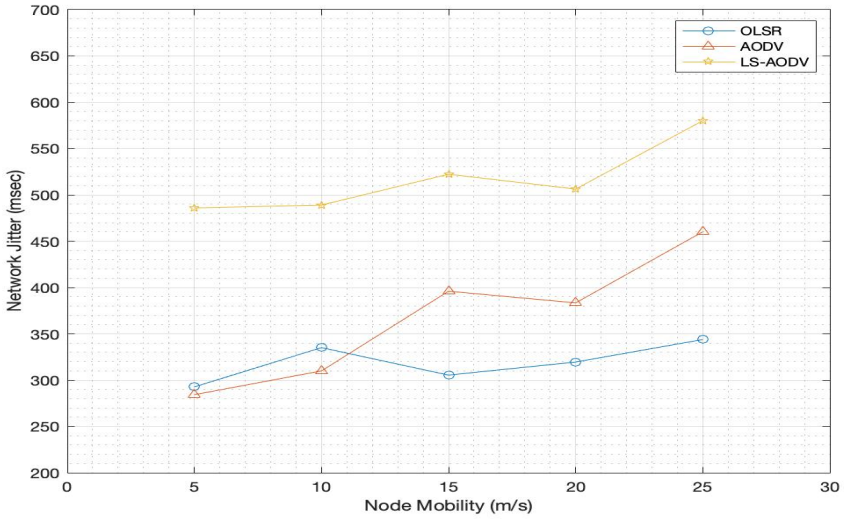

Figure 5.3. Network Jitter at Various Speeds. The Node Speed is Adjusted from 5 m/s to 25 m/s in Increments of 5 m/s.

Figure 5.4 shows the routing overhead as a function of node mobility. Across all simulations, OLSR delivers the highest routing overhead due to the routing protocols design philosophy that requires continuous monitoring and updating of route tables. A key benefit of LS-AODV is that no additional packet types or routing modifications are required to provide authenticity, integrity, and confidentiality. Therefore, as expected, LS-AODV exhibits nearly the same routing overhead as AODV, with only on average 40 more RREQ, RREP, or RERR messages propagating across the network.

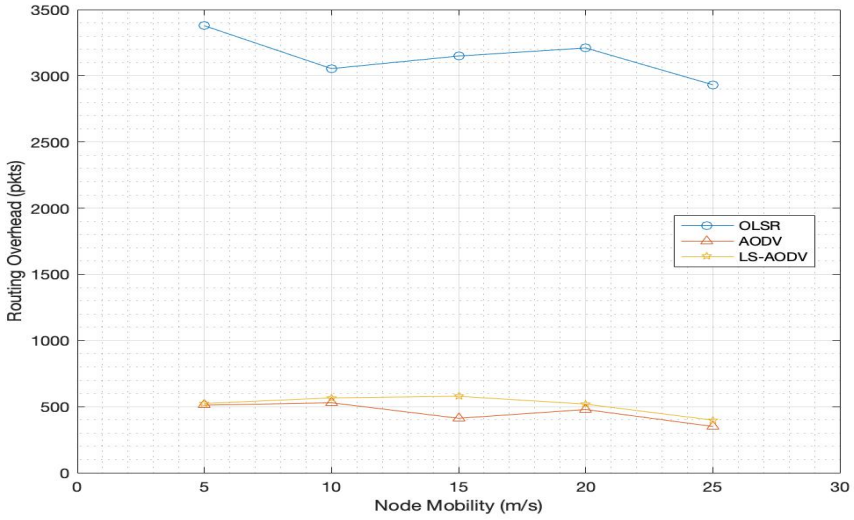

Figure 5.4. Routing Overhead at Various Speeds. The Node Speed is Adjusted from 5 m/s to 25 m/s in Increments of 5 m/s.

5.1.2 Node Density

In this set of simulations, all parameters from Table 5.1 remain the same except for node speed remaining constant at 10 m/s and node density varying from 20 to 100 in increments of 10 nodes. LS-AODV, OLSR, and AODV generally exhibit negative performance trends as the number of nodes in a grid area increases. This simulation set aims to analyze how LS-AODV compares to other routing protocols when node densities scale, not finding an optimal density of nodes.

An important observation comes from analyzing the PDR as the node density varies. Figure 5.5 suggests that LS-AODV outperforms OLSR and AODV with simulations containing 60 nodes or more, the highest positive difference coming at a density of 90 nodes. Furthermore, although the PDR for all three protocols decreases, LS-AODV provides the smallest change in PDR at 13%. Figure 5.6 proposes that in terms of delay, similar to the trend of LS-AODV with varying node speeds, LS-AODV consistently exhibits higher delay regardless of the size of the FANET. LS-AODV produces approximately 50% more delay than AODV

in simulations with sparse grid areas. LS-AODV requires additional packet processing at each intermediate node, reinforcing the expectation that LS-AODV will consistently exhibit higher delay across simulations. LS-AODV closes the gap with AODV and OLSR from 50-90 nodes in denser node populations, nearly matching AODV delay at 100 nodes.

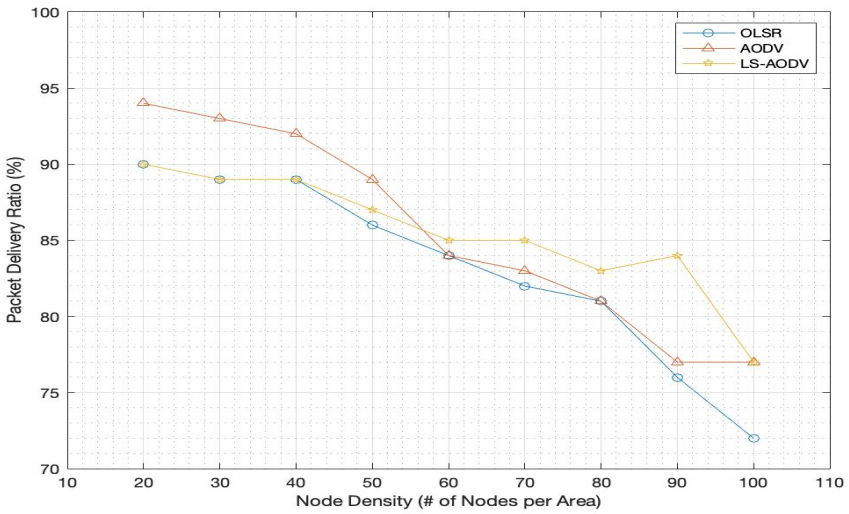

Figure 5.5. Packet Delivery Ratio with Varied Node Density. The Node Density is Adjusted from 20 to 100 in Increments of 10 Nodes.

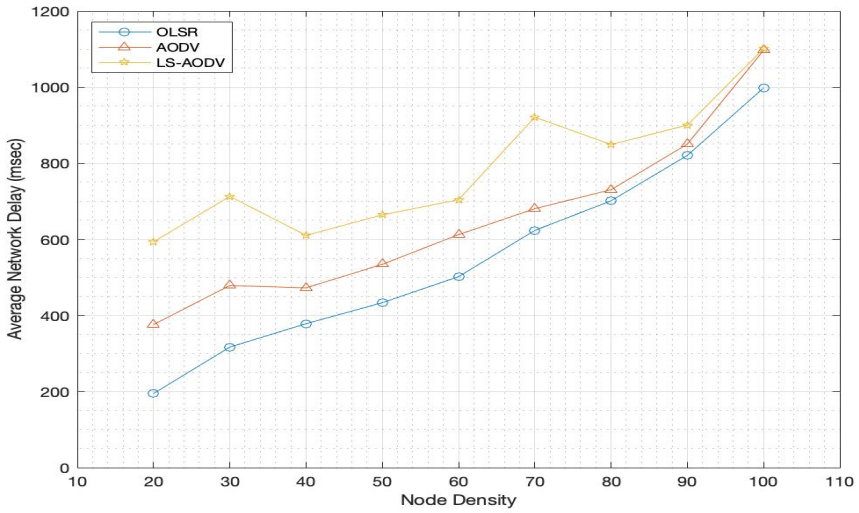

Figure 5.6. Average Network Delay with Varied Node Density. The Node Density is Adjusted from 20 to 100 in Increments of 10 Nodes.

Figures 5.7 and 5.8 describe the jitter and routing overhead for sparse or densely populated FANETs, respectively. Network jitter projects similar results to the delay findings. The routing overhead of AODV and LS-AODV perform considerably better than OLSR as the node density increases. Summarizing the results of FANET performance as node density varies, OLSR, in general, outperforms the other routing protocols in terms of delay and jitter. However, AODV and LS-AODV offer consistently lower routing overhead and higher PDR, with LS-AODV providing improved PDR in densely populated FANETs.

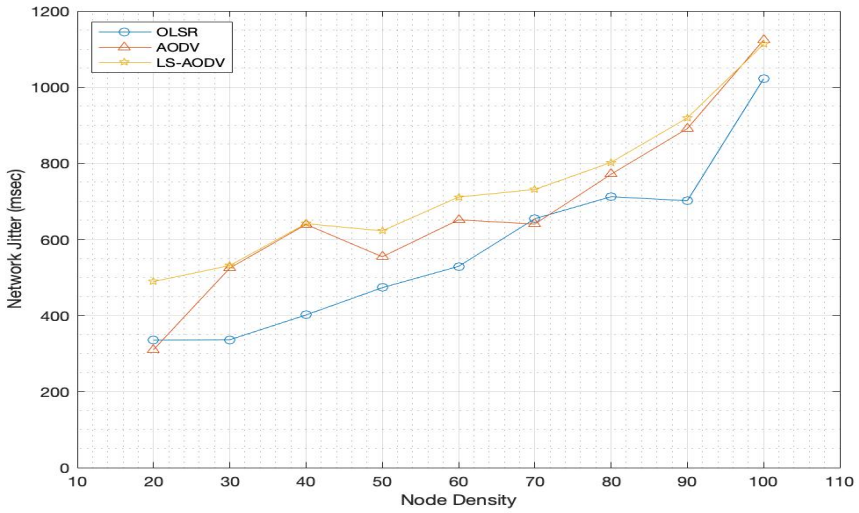

Figure 5.7. Network Jitter with Varied Node Density. The Node Density is Adjusted from 20 to 100 in Increments of 10 Nodes.

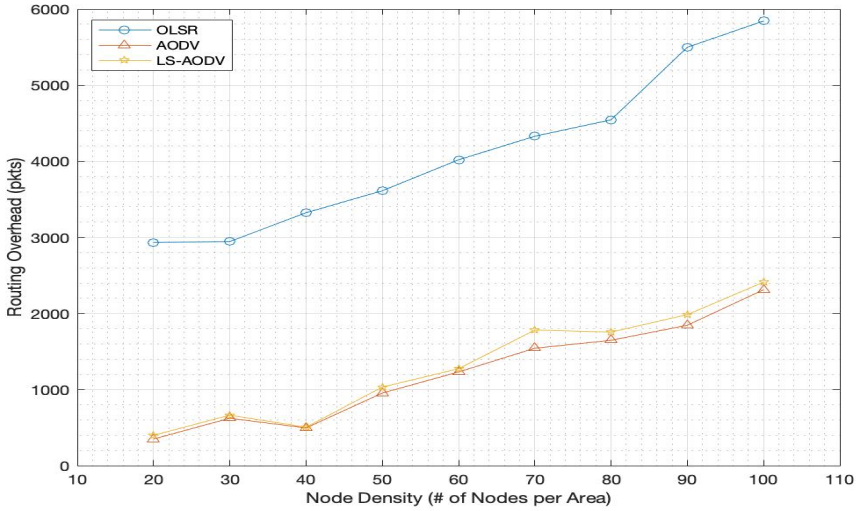

Figure 5.8. Routing Overhead with Varied Node Density. The Node Density is Adjusted from 20 to 100 in Increments of 10 Nodes.

5.1.3 Network Loading

Figures 5.9 - 5.12 depict the effect of network loading on various performance metrics. In all cases, the connection rate (packets/sec) increases from 5 to 30 in increments of 5.

Figure 5.9 indicates that regardless of the chosen routing protocol, the PDR declines in a near-linear fashion. For all connection rates tested, AODV without a cryptographic footprint provides the highest PDR. LS-AODV and OLSR present similar trends, with LS-AODV slightly outperforming OLSR at mid-range connection rates. Also, on average, LS-AODV offers a 3% degrade in PDR across all connections rates.

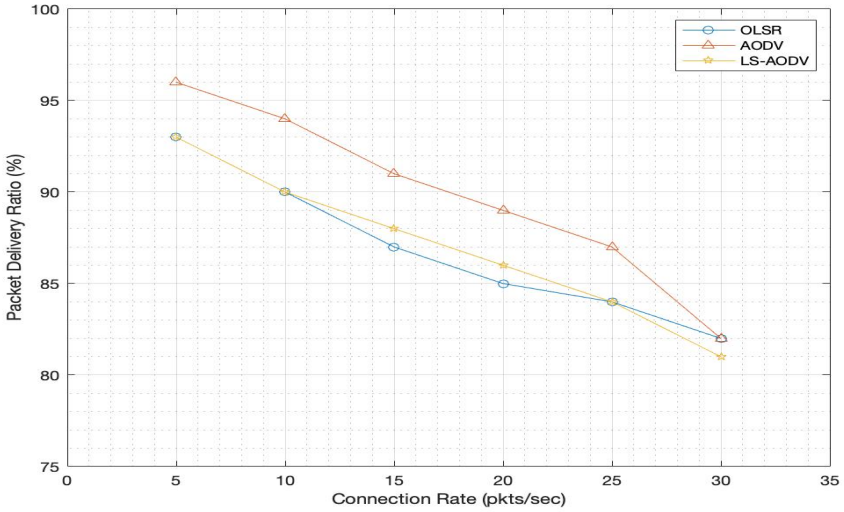

Figure 5.9. Packet Delivery Ratio with Varied Network Loading. The Connection Rate is Adjusted from 5 to 30 pkts/sec in Increments of 5.

Figures 5.10 and 5.11 show the effect of network loading on delay and jitter, respectively. OLSR delivers the lowest delay and jitter results for all network loading conditions. At 20 pkts/sec OLSR and AODV begin to merge, predominately in the delay simulations. LS-AODV induces approximately 210 ms more delay and approximately 187 ms more jitter than AODV. However, LS-AODV provides more consistent delay and jitter results compared to the other routing protocols, with a standard deviation of 31.69 ms in delay and 54.78 ms

in jitter. This consistency reinforces the notion that LS-AODV is advantageous in FANETs with dynamic network conditions.

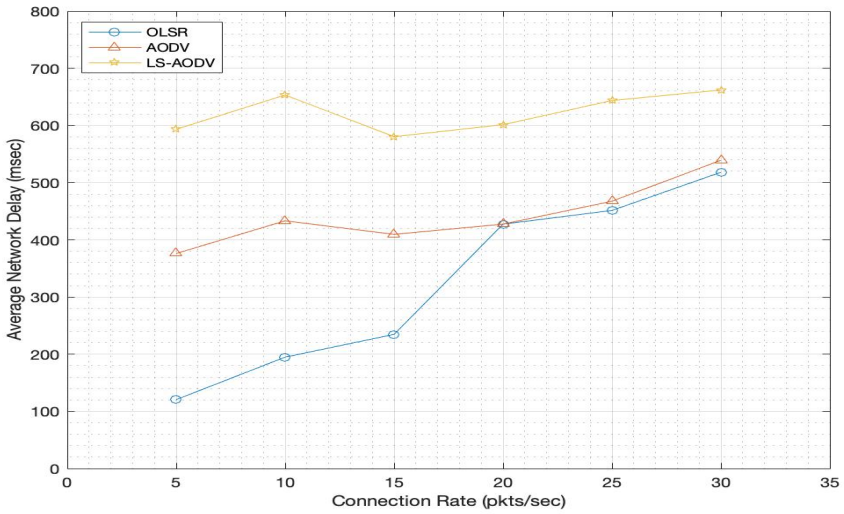

Figure 5.10. Average Network Delay with Varied Network Loading. The Connection Rate is Adjusted from 5 to 30 pkts/sec in Increments of 5.

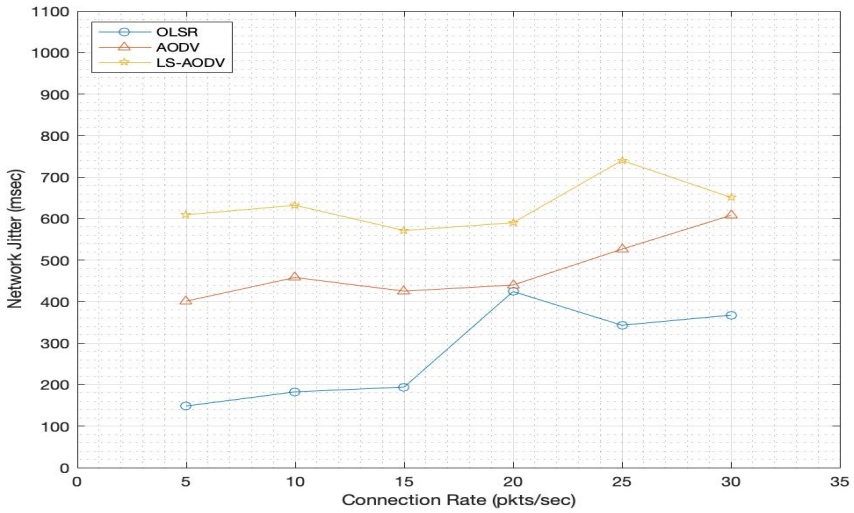

Figure 5.11. Network Jitter with Varied Network Loading. The Connection Rate is Adjusted from 5 to 30 pkts/sec in Increments of 5.

The routing overhead is significantly affected by the connection rate. In Figure 5.12, we observe that with the AODV and LS-AODV routing protocols, the routing overhead increases nearly fourfold when the connection rate changes from 5 pkts/sec to 30 pkts/sec. OLSR delivers relatively consistent yet high routing overhead results regardless of the connection rate. As discussed in previous simulation sections, the expectation is that LS-AODV demands nearly the same overhead footprint as the baseline AODV protocol. However, a peculiarity exists at a connection rate of 25 pkts/sec; LS-AODV requires 305 more routing packets for path discovery and maintenance than AODV requires.

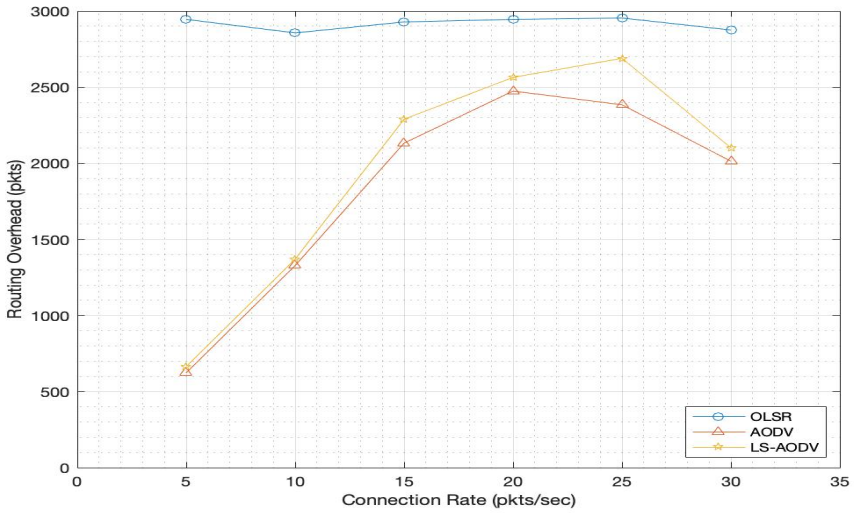

Figure 5.12. Routing Overhead with Varied Network Loading. The Connection Rate is Adjusted from 5 to 30 pkts/sec in Increments of 5.

5.1.4 Pause Time

Figures 5.13 - 5.16 depict the effects of pause time on various performance metrics. In all cases, the pause time (secs) increases from 5 to 30 in increments of 5. In a military-oriented FANET using nano drones, the pause time may vary depending on the mission set and environmental conditions. For instance, if a swarm of nano drones follows a moving vehicle, the pause time may be minimal. However, the opposite case may arise when the nano drones conduct surveillance operations without a specific target. Once a drone acquires a target of interest, it may pause in place for a variable amount of time.

Figure 5.13 illustrates the effect of pause time on the PDR. While the results are similar to those found in the variable mobility case, a few key distinctions are evident. Overall, changes in pause time produces slightly higher variability in PDR than changes in node mobility. For example, the PDR standard deviations for AODV and LS-AODV during the mobility simulation were 1.33 and 1.02, respectively. On the contrary, in the pause time simulation, the PDR standard deviations for AODV and LS-AODV are 1.67 and 1.24. However, the

highest PDRs out of any simulation for AODV and LS-AODV result when the pause time is a moderate 15 seconds. Compared to AODV and OLSR, the LS-AODV routing protocol delivers slightly higher marks for PDR when the pause time is 10 or 30 seconds.

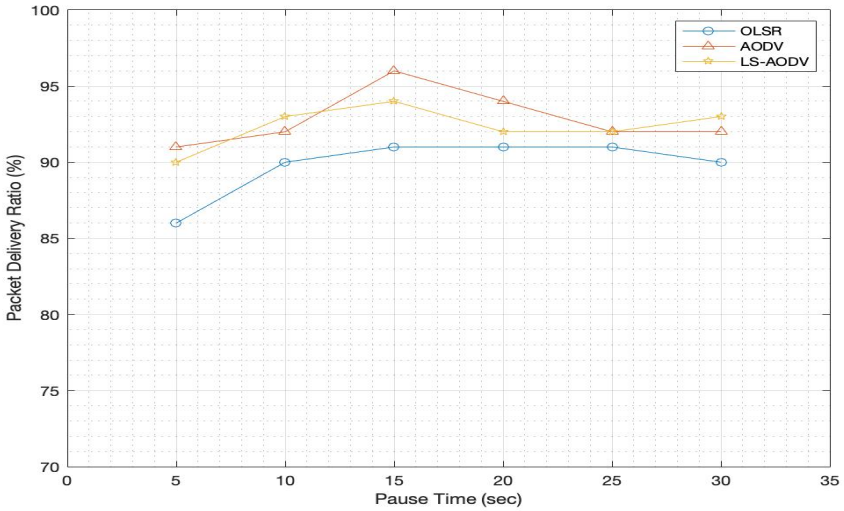

Figure 5.13. Packet Delivery Ratio with Varied Pause Time. The Pause Time is Adjusted from 5 to 30 seconds in Increments of 5 Seconds.

The delay and jitter simulations produce similar trends. In Figures 5.14 and 5.15, we notice a significant spike in delay and jitter at a pause time of ten seconds for all three protocols. LS-AODV shows significant improvement when subject to simulations with higher pause times. Notably, at pause times greater than 15 seconds, LS-AODV delivers a slightly higher delay than AODV, and at pause times greater than 20 seconds produces slightly less jitter at each node. The routing overhead comparison in Figure 5.16 illustrates that AODV and LS-AODV consistently require fewer routing control packets than OLSR. With a 10-second pause time, OLSR requires 3512 control packets while AODV and LS-AODV require 481 and 521, respectively.

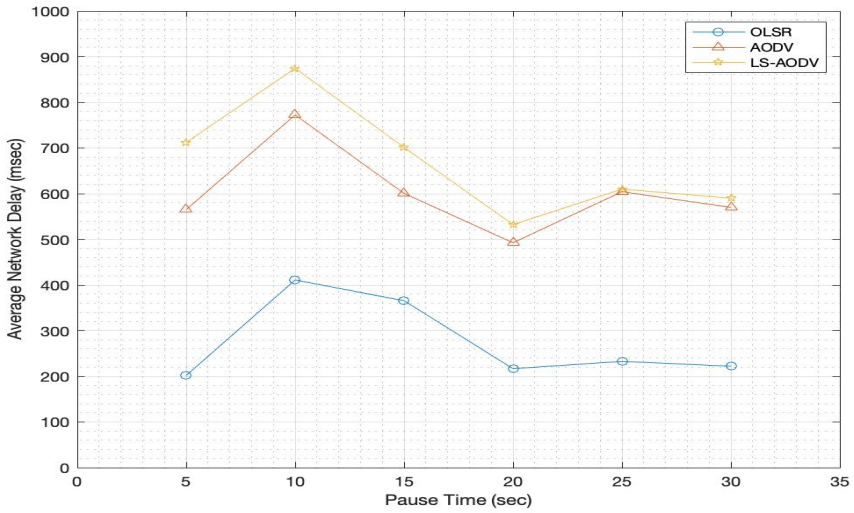

Figure 5.14. Average Network Delay with Varied Pause Time. The Pause Time is Adjusted from 5 to 30 Seconds in Increments of 5 Seconds.

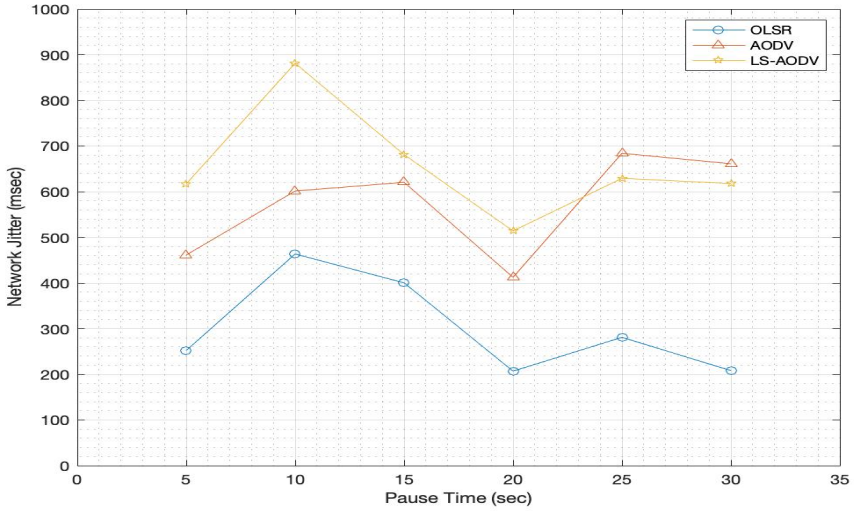

Figure 5.15. Network Jitter with Varied Pause Time. The Pause Time is Adjusted from 5 to 30 Seconds in Increments of 5 Seconds.

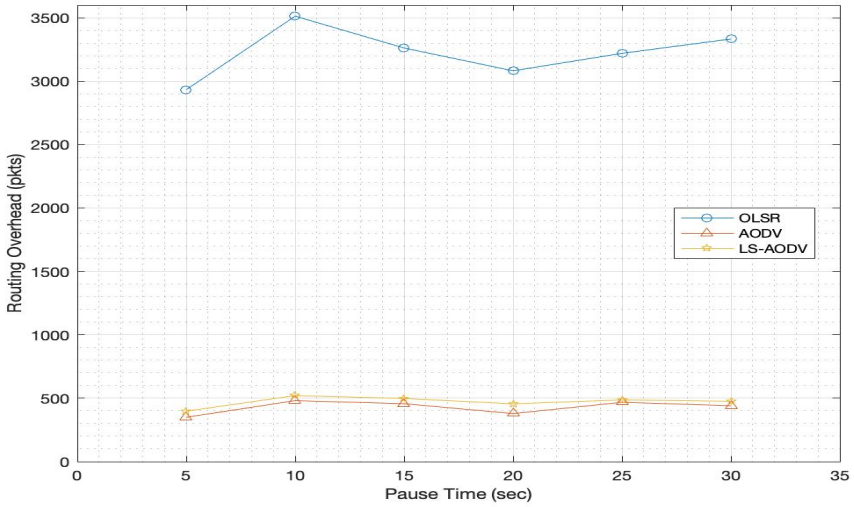

Figure 5.16. Routing Overhead with Varied Pause Time. The Pause Time is Adjusted from 5 to 30 Seconds in Increments of 5 Seconds.

5.1.5 Throughput - Nominal Conditions

This section sets the conditions to evaluate the throughput available while conducting a specific military operation. A FANET of nano drones works cooperatively to track and deploy signal disruption capabilities above a fast-moving vehicular target. The vehicle is traveling at approximately 60 mph across multiple city blocks. The drones are distributed randomly across the grid and share information regarding the target with the network. Each routing protocol is subject to the same mobility and traffic parameters, detailed in Table 5.2.

Table 5.2. Simulation Parameters for Urban Military Operation

Mobility	
Number of Nodes	40
Map Size	670 m × 670 m
Mobility Model	Random Waypoint
Pause Time	2 sec
Speed	25 m/s
Simulation Time	1000 sec
Traffic	
Traffic Type	Constant Bit Rate
Packet Size	512 bytes
Connection Rate	10 pkts/sec
Routing Protocol	
Protocols	OLSR, AODV, LS-AODV

Figure 5.17 highlights the differences in throughput availability throughout the roughly 17-minute mission. This simulation aims to provide one type of scenario that briefly highlights the capability of LS-AODV in terms of network strength relative to two prominent FANET routing protocols. OLSR provides the highest marks for throughput at 2.55 Mbps, and LS-AODV delivers the lowest throughput at 2.00 Mbps. At the end of the next section of simulations, the same scenario is put forth but with the realistic addition of malicious nodes attempting to disrupt the military-based support FANET. With OLSR and AODV implementing no security architecture to protect routing control packets, LS-AODV should drastically outperform either routing protocol.

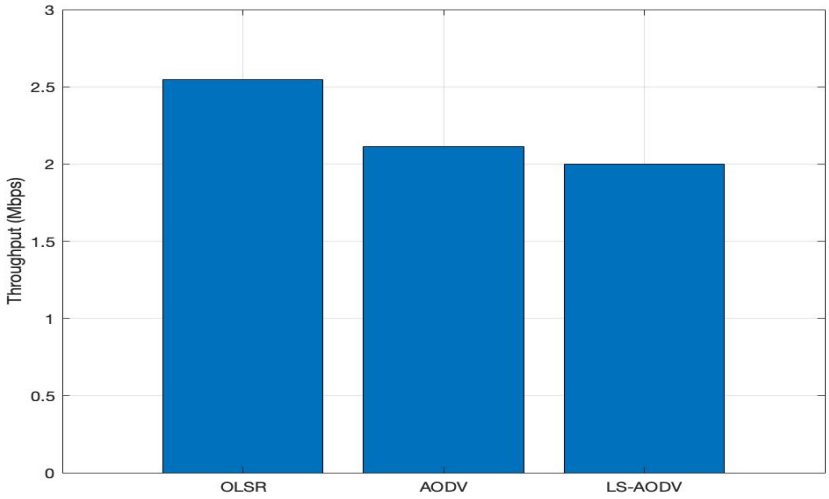

Figure 5.17. Throughput Available for an Urban Military Operation. The FANET of Nano Drones Works Cooperatively to Track and Deploy Signal Disruption Capabilities Above a Fast-Moving Vehicular Target.

5.2 Black Hole Attack FANET Scenario

In this set of simulations, the goal is to examine the performance of a FANET of nano drones while under attack from a nefarious actor. Specifically, the network is subject to a black-hole attack with a range of malicious nodes.

5.2.1 Black-Hole Attack in AODV

An approved and vetted patch for several versions of NS-3 exists that implements the black-hole attack within the AODV routing protocol. However, a similar patch is not available for the OLSR routing protocol. Therefore, this scenario compares LS-AODV and AODV while subject to a black-hole attack.

During the AODV route discovery process, a black-hole node convinces the source node that it can provide the shortest path to a destination node. Specifically, the black-hole node

responds to the source node RREQ with a false RREP containing the highest destination sequence number. The source node will use the first reply from an intermediate node to establish the route to the destination node. If a normal node responds first to the RREQ, then the protocol works as intended. However, since the black-hole node does not need to check its routing table during operation, it's highly likely that its false RREP will arrive at the source node first [19]. If the source node decides to use the route from the black-hole node, all packets arriving at the black-hole node will be discarded. Without a robust cryptographic footprint to protect the routing packets, the AODV routing protocol is easily susceptible to a black-hole attack with little effort from a nefarious operator.

5.2.2 Mobility Under Threat

This simulation case investigates how changing the node speed affects routing protocol performance while under threat from a black-hole attack. This simulation uses the same mobility and traffic model parameters implemented in the non-threat FANET scenario of Section 5.1. Two randomly chosen nodes within the simulation carry out the black-hole attack. This corresponds to 10% of the FANET being compromised during the simulation run. Table 5.3 summarizes the simulation parameters for the mobility case.

Table 5.3. Simulation Parameters for Network Under Threat

Mobility	
Number of Nodes	20
Map Size	670 m × 670 m
Mobility Model	Random Waypoint
Pause Time	2 sec
Speed	10 m/s
Simulation Time	200 sec
Traffic	
Traffic Type	Constant Bit Rate
Packet Size	512 bytes
Connection Rate	10 pkts/sec
Routing Protocol	
Protocols	AODV, LS-AODV
Number of Malicious Nodes	2

Figure 5.18 shows the PDR as a function of node mobility, with normal and malicious nodes moving randomly across the grid area. Across the range of mobilities, LS-AODV outperforms AODV. Specifically, LS-AODV offers on average 23.8% higher PDR then AODV. Furthermore, as similarly shown in the mobility study of Section 5.1, LS-AODV provides more consistent PDR as the node speed changes. These results are expected because LS-AODV mitigates the effects of the black-hole attack by using a lightweight cryptographic footprint.

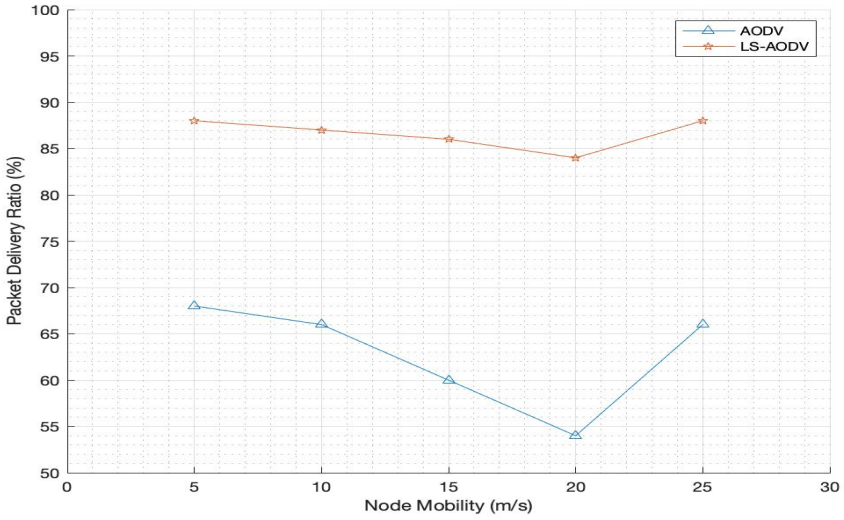

Figure 5.18. Packet Delivery Ratio at Various Speeds (Black-Hole Attack). The Node Speed is Adjusted from 5 m/s to 25 m/s in Increments of 5 m/s.

Figures 5.19, 5.20, and 5.21 depict the delay, jitter, and routing overhead, respectively. At all node speeds besides 15 m/s, AODV induces less delay, and regardless of the chosen node speed, the jitter sum and routing overhead for AODV remains consistently lower than the jitter sum and routing overhead for LS-AODV. There are two potential reasons for this result. One is that LS-AODV requires more processing time at each intermediate node for encrypting and authenticating each routing packet, therefore inducing higher delay, jitter, and routing overhead. A second reason is that because the simulation does not distinguish between normal nodes and black-hole nodes during the calculation of performance metrics when a packet arrives at a black-hole node, its path terminates, resulting in artificially low-performance metric readouts in the AODV case.

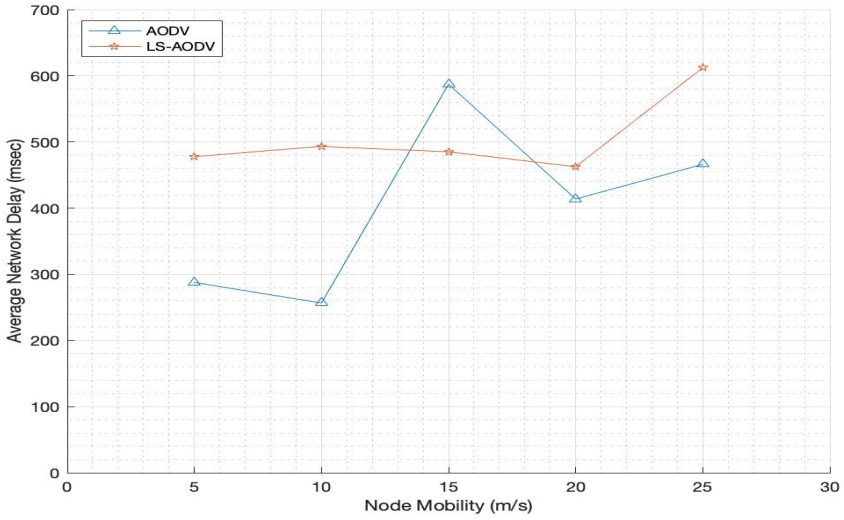

Figure 5.19. Average Network Delay at Various Speeds (Black-Hole Attack). The Node Speed is Adjusted from 5 m/s to 25 m/s in Increments of 5 m/s.

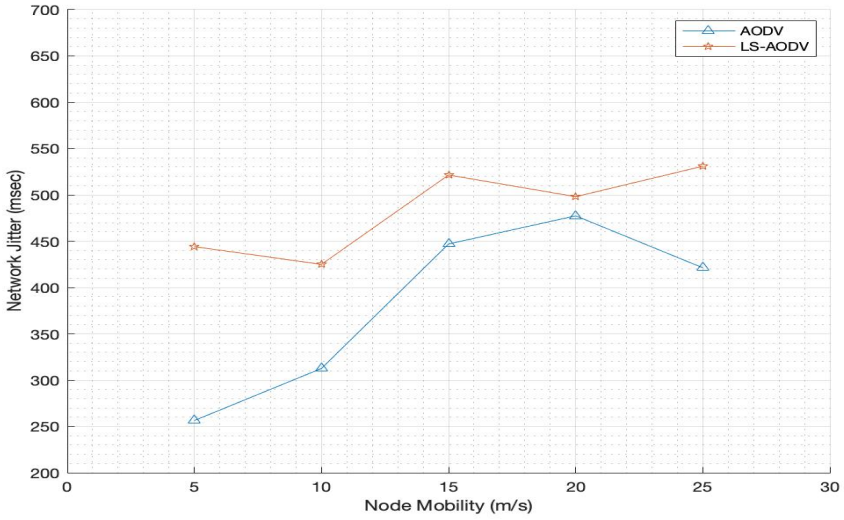

Figure 5.20. Network Jitter at Various Speeds (Black-Hole Attack). The Node Speed is Adjusted from 5 m/s to 25 m/s in Increments of 5 m/s.

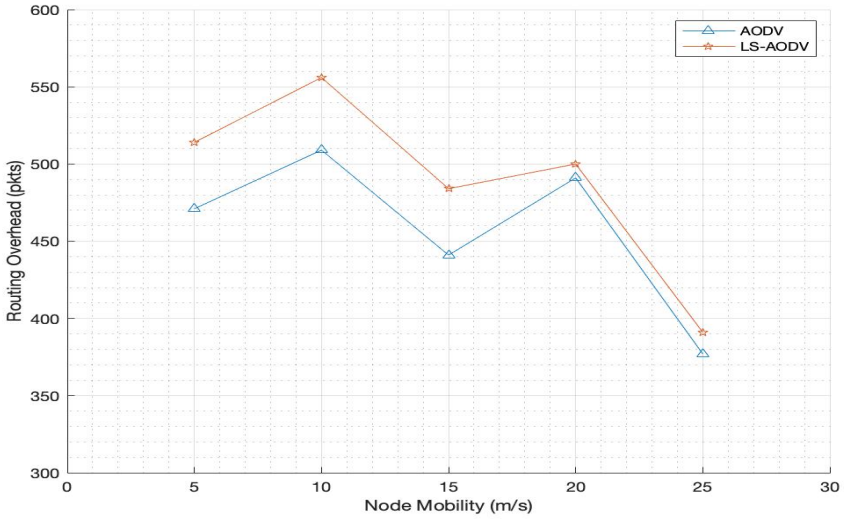

Figure 5.21. Routing Overhead at Various Speeds (Black-Hole Attack). The Node Speed is Adjusted from 5 m/s to 25 m/s in Increments of 5 m/s.

5.2.3 Node Density Under Threat

In this set of simulations, all parameters from Table 5.3 remain the same except for the node speed remaining constant at 10 m/s and node density varying from 20 to 100 in increments of 10 nodes. Also, the number of black-hole nodes increases as the node density increases to provide a constant 10% malicious-to-normal node ratio. The goal is to analyze how LS-AODV compares to AODV when node densities scale and the routing protocols contend with random black-hole nodes scattered throughout the grid area.

Unlike the node density simulation in Section 5.1, where LS-AODV only delivers higher PDR in denser networks, Figure 5.22 suggests that LS-AODV outperforms AODV at all node density levels. Furthermore, although the PDR for both protocols decreases, LS-AODV provides the smallest change in PDR at 17%, while the AODV PDR decreases by 25%. In Figures 5.23, 5.24, and 5.25, the delay, jitter, and routing overhead during a black-hole attack is shown. These figures reinforce the notion that LS-AODV continues to require higher delay, jitter, and routing overhead regardless of the inclusion of black-hole nodes in

the network. However, a few exceptions are present when analyzing the jitter sum results. In networks with 40, 80, or 90 nodes (10% of which are malicious), the jitter sum for LS-AODV is slightly lower. The results of the node density simulation highlight the strength of LS-AODV in providing consistently high PDR as the network scalability changes.

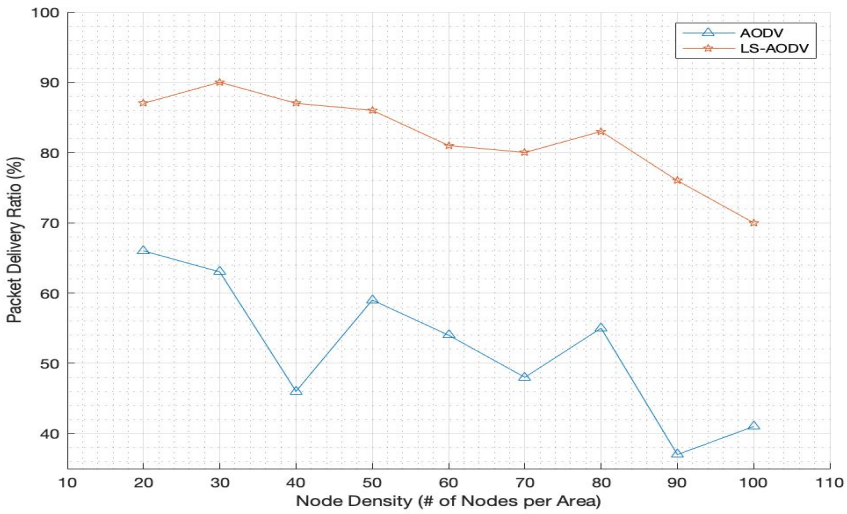

Figure 5.22. Packet Delivery Ratio with Varied Node Density (Black-Hole Attack). The Node Density is Adjusted from 20 to 100 in Increments of 10 Nodes.

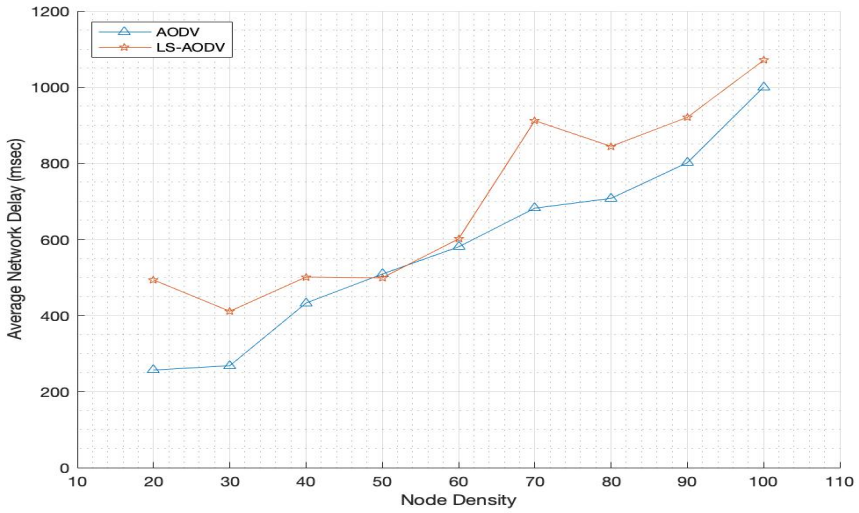

Figure 5.23. Average Network Delay with Varied Node Density (Black-Hole Attack). The Node Density is Adjusted from 20 to 100 in Increments of 10 Nodes.

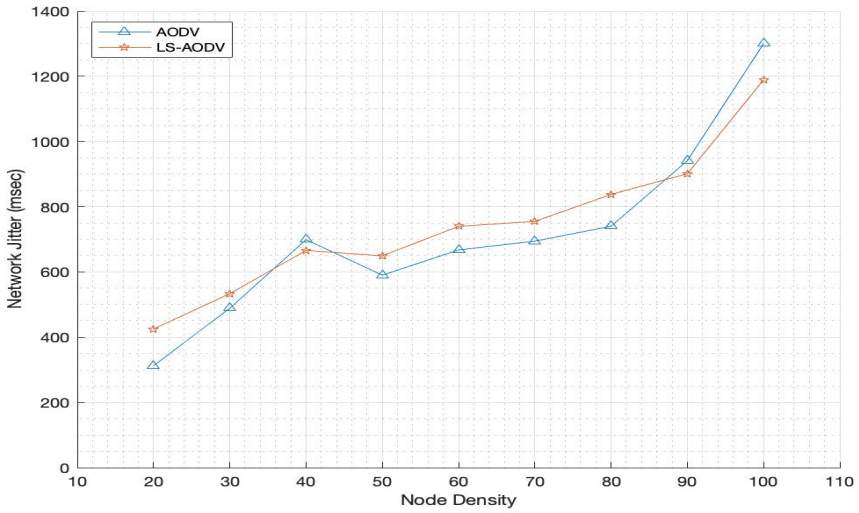

Figure 5.24. Network Jitter with Varied Node Density (Black-Hole Attack). The Node Density is Adjusted from 20 to 100 in Increments of 10 Nodes.

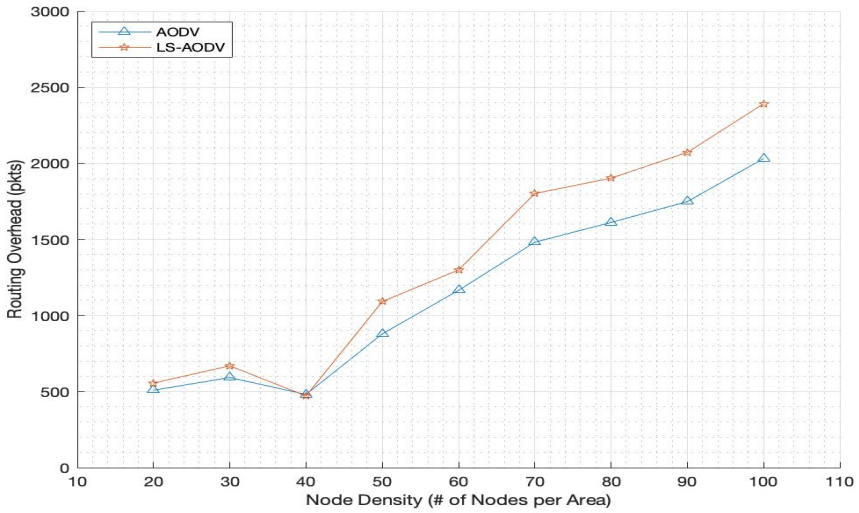

Figure 5.25. Routing Overhead with Varied Node Density (Black-Hole Attack). The Node Density is Adjusted from 20 to 100 in Increments of 10 Nodes.

5.2.4 Network Loading Under Threat

This simulation set depicts the effects of network loading in a black-hole network on various performance metrics. In all cases, the parameters match those of Table 5.3 except that the connection rate (packets/sec) increases from 5 to 30 in increments of five, and the node speed is set at ten m/s.

Figure 5.26 shows the PDR for both routing protocols. LS-AODV continues to provide higher PDR compared to AODV regardless of the changes made to the network parameters. Note the near-linear progression of the PDR of LS-AODV from 5 to 30 pkts/sec. While a negative trend in PDR is present, the linear fashion of the LS-AODV routing protocol suggests some amount of predictability is present as the connection rate changes. Further analysis will provide greater insight into the predictive nature of the LS-AODV routing protocol.

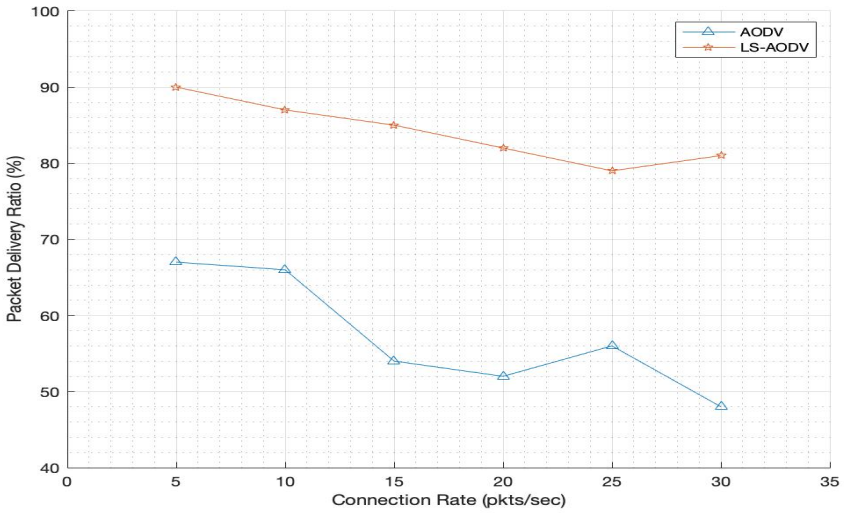

Figure 5.26. Packet Delivery Ratio with Varied Network Loading (Black-Hole Attack). The Connection Rate is Adjusted from 5 to 30 pkts/sec in Increments of 5.

Figures 5.27 - 5.29 depict the remaining performance metrics. The results show similar trends to those in Section 5.1 when the networks only contain normal nodes. However, with the introduction of black-hole nodes, more significant fluctuations exist in the delay and jitter results for either routing protocol. A spike in the delay and jitter for LS-AODV exists at higher connection rates (> 20 pkts/sec). Since the LS-AODV routing protocol avoids the black-hole nodes during the route discovery process, fewer valid network nodes are available to transport packets from source to destination. The reduction in viable routes means that as the connection rate rises, bottlenecks may form in the queues of particular intermediate nodes.

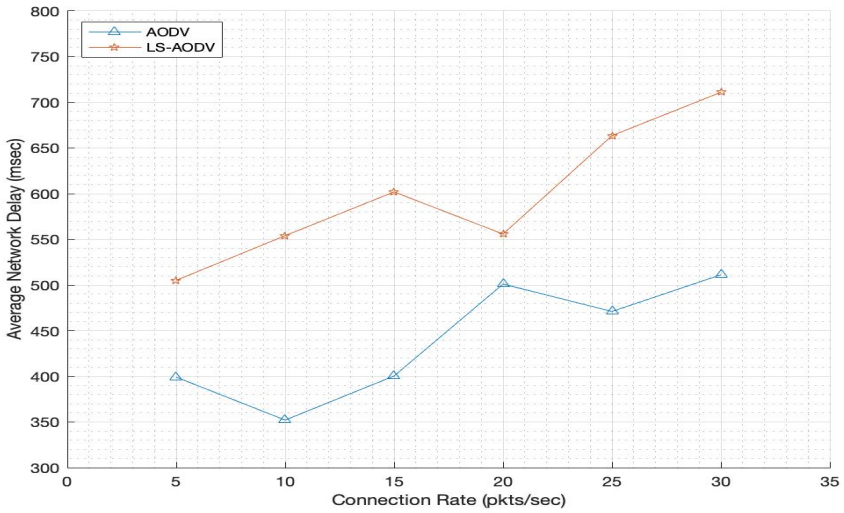

Figure 5.27. Average Network Delay with Varied Network Loading (Black-Hole Attack). The Connection Rate is Adjusted from 5 to 30 pkts/sec in Increments of 5.

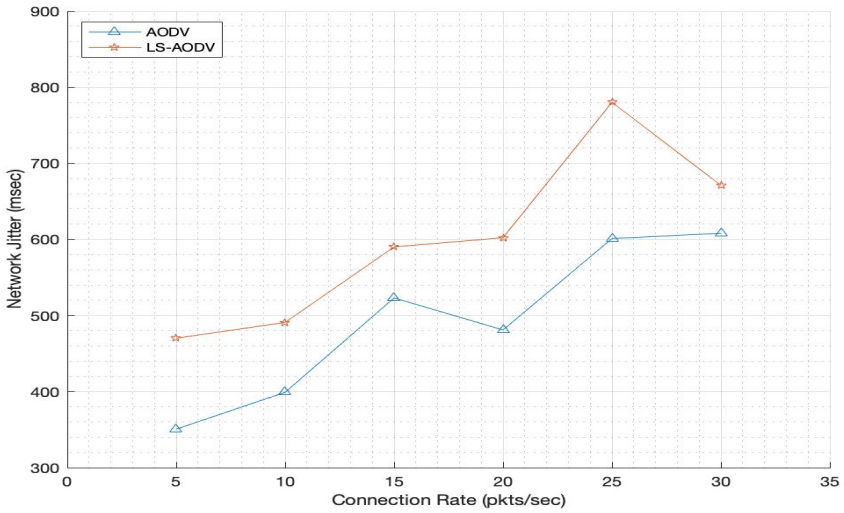

Figure 5.28. Network Jitter with Varied Network Loading (Black-Hole Attack). The Connection Rate is Adjusted from 5 to 30 pkts/sec in Increments of 5.

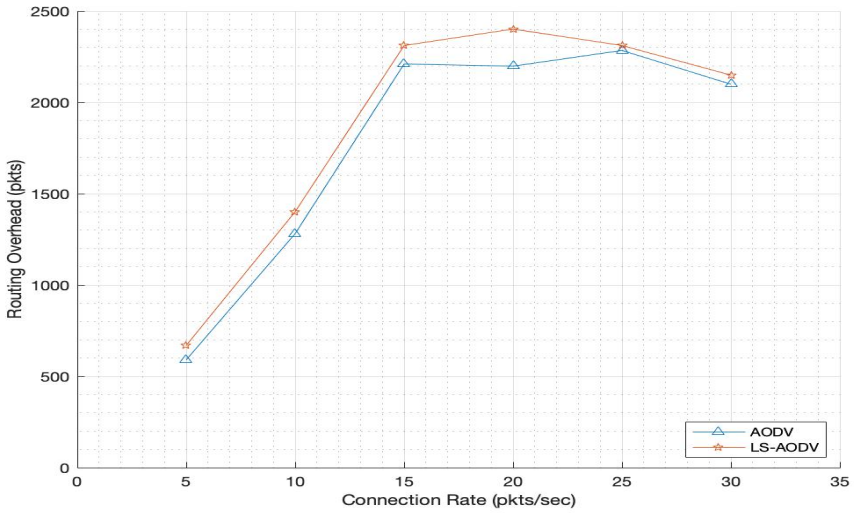

Figure 5.29. Routing Overhead with Varied Network Loading (Black-Hole Attack). The Connection Rate is Adjusted from 5 to 30 pkts/sec in Increments of 5.

5.2.5 Pause Time Under Threat

Figures 5.30 - 5.33 depict the effects of pause time in a black-hole network on various performance metrics. In all cases, the pause time (secs) increases from 5 to 30 in increments of 5.

Figure 5.30 illustrates the effect of pause time on the PDR. The standard deviation of the LS-AODV PDR is 1.49%, while the standard deviation of the AODV PDR is 5.40%. LS-AODV provides higher and more consistent PDR than AODV regardless of the chosen pause time. Figures 5.31 and 5.32 highlight the delay and jitter as the pause time varies. The inclusion of black-hole nodes in the network causes a significant degree of fluctuation and uncertainty in delay and jitter for either routing protocol. While LS-AODV continues to induce higher delay and jitter in the network, specific cost optimization techniques may reduce this negative trend across all network types.

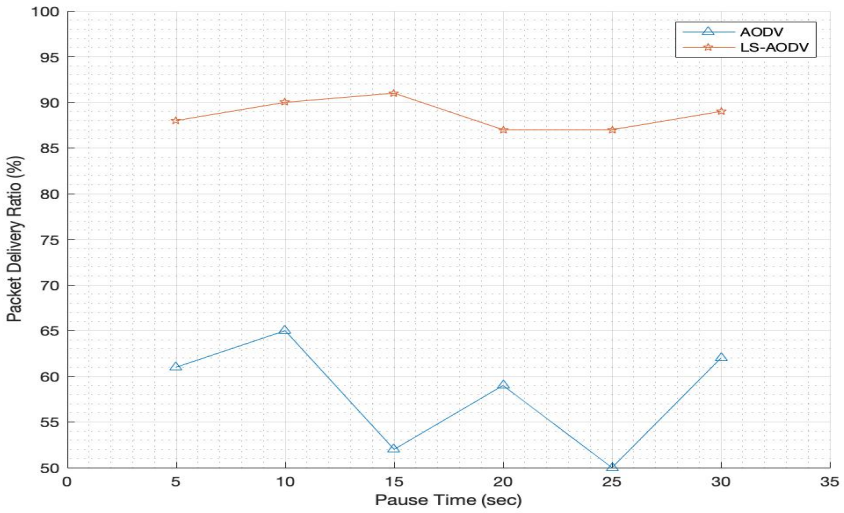

Figure 5.30. Packet Delivery Ratio with Varied Pause Time (Black-Hole Attack). The Pause Time is Adjusted from 5 to 30 seconds in Increments of 5 Seconds.

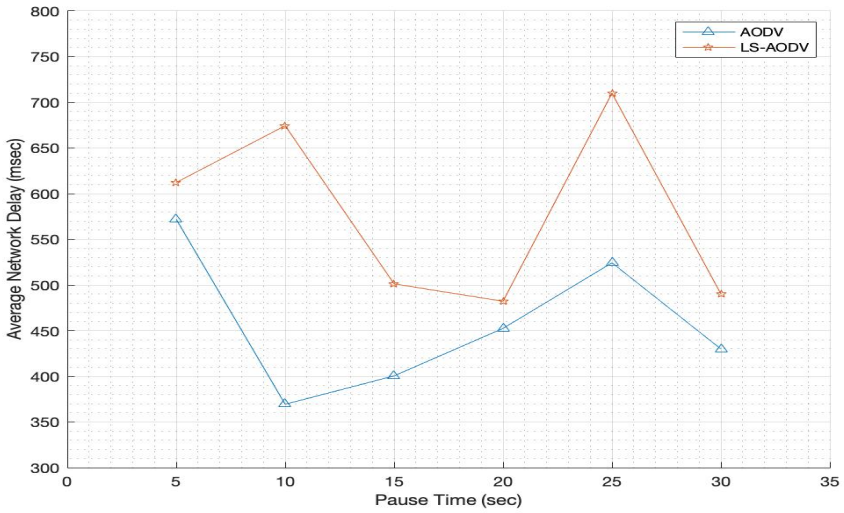

Figure 5.31. Average Network Delay with Varied Pause Time (Black-Hole Attack). The Pause Time is Adjusted from 5 to 30 seconds in Increments of 5 Seconds.

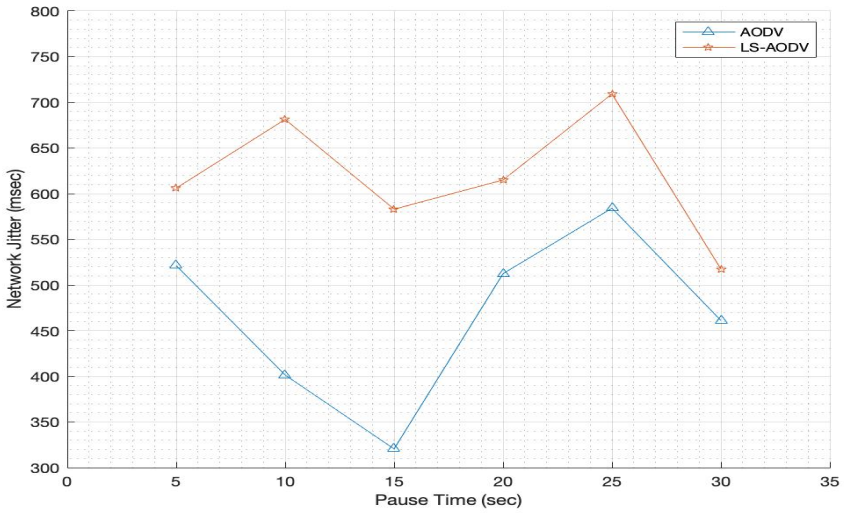

Figure 5.32. Network Jitter with Varied Pause Time (Black-Hole Attack). The Pause Time is Adjusted from 5 to 30 seconds in Increments of 5 Seconds.

Figure 5.33 highlights the effect of pause time on routing overhead when black-hole nodes are randomly dispersed throughout the network. At the lower and higher ends of the pause time spectrum, LS-AODV requires, on average, 57 more packets to establish and maintain routes throughout the network. However, at median pause times, LS-AODV requires, on average, 170 more packets.

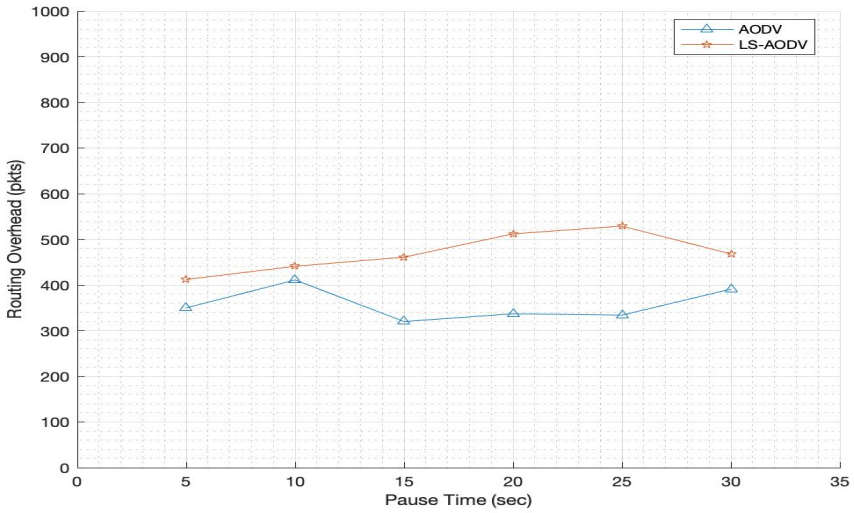

Figure 5.33. Routing Overhead with Varied Pause Time (Black-Hole Attack). The Pause Time is Adjusted from 5 to 30 seconds in Increments of 5 Seconds.

5.2.6 Malicious Node Density

This simulation describes the throughput availability as the number of malicious nodes changes. Twenty nodes are randomly spread out across a grid, with 1–5 of those nodes being designated as black-holes.

Figure 5.34 shows that as black-hole nodes compromise a larger portion of the FANET, the throughput availability for LS-AODV and AODV diminishes. However, a significant negative trend difference exists between both routing protocols. Specifically, the throughput of LS-AODV reduces by 0.54 Mbps, while the throughput of AODV reduces by 1.17 Mbps. Also, note that LS-AODV provides higher throughput, particularly in the extreme case where 25% of the FANET is compromised. AODV only provides 0.29 Mbps versus LS-AODV at 1.54 Mbps, an 81.0% reduction in throughput availability.

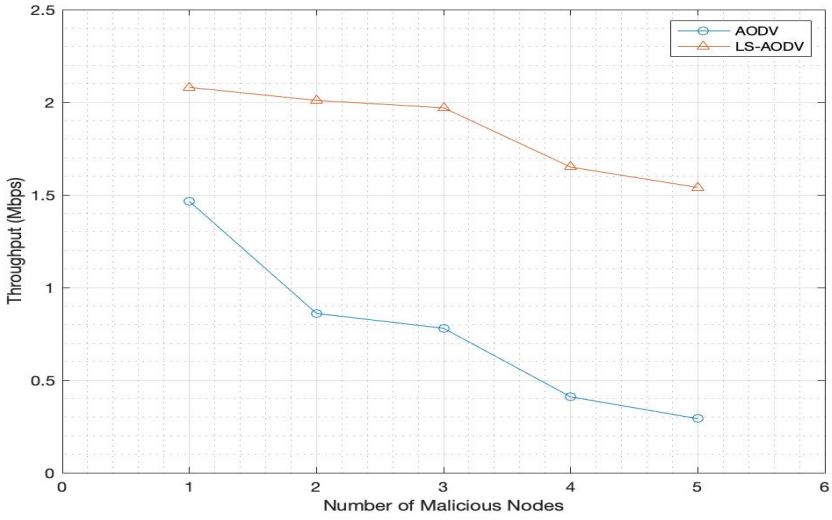

Figure 5.34. Throughput with Varied Number of Malicious Nodes.

5.2.7 Throughput - Threat Conditions

This section sets the conditions to evaluate the throughput available while conducting a specific military operation in a combat zone with known threats. A FANET of nano drones works cooperatively to track and deploy signal disruption capabilities above a fast-moving vehicular target. The vehicle is traveling at approximately 60 mph across multiple city blocks. The drones are distributed randomly across the grid and share information regarding the target with the network. The enemy combatants attempt to disrupt the FANET by deploying five nano drones tasked with conducting a black-hole attack. Each routing protocol is subject to the same mobility and traffic parameters, detailed in Table 5.4.

Table 5.4. Simulation Parameters for Urban Military Operation Under Threat

Mobility	
Number of Nodes	40
Map Size	670 m × 670 m
Mobility Model	Random Waypoint
Pause Time	2 sec
Speed	25 m/s
Simulation Time	1000 sec
Traffic	
Traffic Type	Constant Bit Rate
Packet Size	512 bytes
Connection Rate	10 pkts/sec
Routing Protocol	
Protocols	AODV, LS-AODV
Number of Malicious Nodes	5

Figure 5.35 highlights the difference in throughput availability throughout the roughly 17-minute mission. With only 12.5% of the FANET compromised by black-hole nodes, the throughput available by drones using the AODV routing protocol is significantly hampered. AODV provides 0.519 Mbps throughput compared to the optimal case scenario in Section 5.1.5, where AODV provides 2.111 Mbps, a 75.4% reduction. With such a considerable reduction in throughput, the drones using AODV may be unable to effectively track and disrupt the communication signal of the vehicle of interest. On the other hand, LS-AODV performs exceptionally well in comparison to AODV. LS-AODV delivers 1.842 Mbps throughput, a 254.9% greater capacity. Also, the LS-AODV routing protocol, compared to the optimal case scenario in Section 5.1.5, provides only 7.8% less throughput when operating with malicious nodes attempting to disrupt the protocol's functionality.

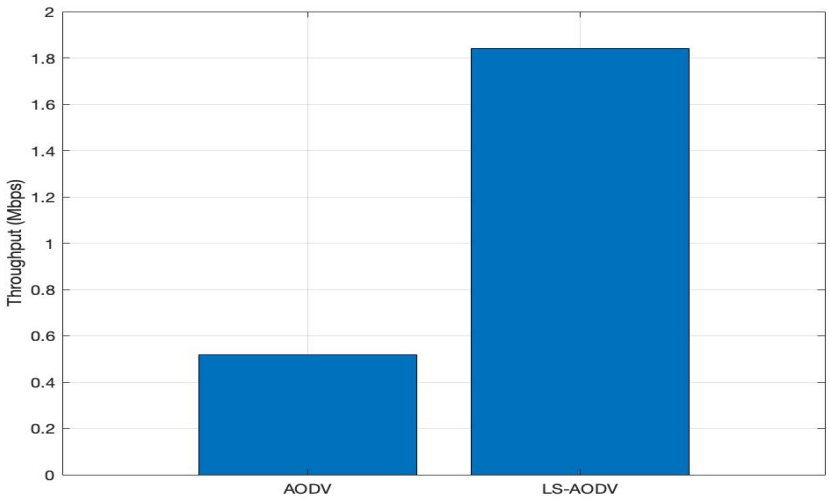

Figure 5.35. Throughput Available for an Urban Military Operation Under Threat. 12.5% of the FANET is Compromised by Black-Hole Nodes.

5.3 Summary of Results

This chapter conducts simulations based on two FANET scenarios. The first scenario compares LS-AODV to other prominent ad hoc routing protocols when the network operates under optimal conditions, i.e., no black-hole nodes attempting to disrupt communication. The second scenario compares LS-AODV to the AODV routing protocol when the network is subject to persistent black-hole attacks. In both scenarios, various performance metrics are compared while the node mobility, node density, network loading, and pause time changes.

In the first scenario, with optimal conditions, LS-AODV generally provides slightly lower PDR than AODV, higher delay and jitter then both AODV and OLSR, and higher routing overhead than AODV but significantly less overhead than OLSR. However, in a few instances, LS-AODV does outperform or match the performance of the other routing protocols. Specifically, when analyzing the PDR as the node speed changes, LS-AODV outperforms OLSR in all instances and matches the AODV PDR at node speeds of 20 m/s. Also, when analyzing the PDR as the node density changes, LS-AODV performs better than

AODV at higher node densities, with LS-AODV delivering 7% more packets than AODV and 8% more packets than OLSR in FANET simulations with 90 nodes. While reviewing all simulations, a key takeaway is that LS-AODV tends to deliver more consistent and less fluctuating performance metrics, in particular the PDR of the various routing protocols.

In the second scenario, with black-hole nodes included in the FANET, LS-AODV consistently provides a higher PDR regardless of changes made to the network parameters. Across all simulations where black-hole nodes compromise 10% of the FANET, LS-AODV delivers on average 36.2% higher PDR than AODV. The expectation is that LS-AODV will induce more delay and jitter into the FANET, regardless of the FANET setup or inclusion of black-hole nodes. However, in a few instances, LS-AODV produces less delay or jitter than AODV. For example, the set of simulations analyzing mobility indicates that the delay for LS-AODV is 102.13 ms less than that of AODV. Furthermore, in the set of simulations analyzing node density, the jitter value for LS-AODV is lower than that of AODV at node densities of 40, 90, and 100.

In both scenarios, a simulation of realistic mission parameters pits different routing protocols against each other. Precisely, the simulations measure the throughput availability for each routing protocol when a FANET of nano drones is tasked with tracking and disrupting the cell phone signal coming from a moving vehicle. In the first scenario, OLSR provides the highest throughput availability of 2.55 Mbps while AODV and LS-AODV provide 2.11 Mbps and 2.00 Mbps, respectively. In contrast, LS-AODV drastically outperforms AODV when the same scenario is run in Section 5.2, with the only addition being that of 5 black-hole nodes tasked with disrupting the communication links between nano drones. LS-AODV delivers 1.842 Mbps of throughput, a 254.9% greater capacity than AODV under the same conditions. Furthermore, LS-AODV experiences a 0.156 Mbps decline in throughput availability when black-hole nodes are randomly dispersed across the FANET, whereas AODV experiences a 1.592 Mbps decline in throughput availability. Based on the totality of results presented, we have shown that LS-AODV provides a means for secure routing in a FANET of nano drones while not incurring a comparatively higher cost in delay, jitter, or routing overhead.

CHAPTER 6:
Conclusions

This chapter highlights the key findings from the thesis and provides recommendations for future work.

6.1 Summary

Deploying a FANET of nano drones equipped with monitoring or signal-jamming applications ushers in a unique hybrid style of warfare. With a likely low production cost, a large-scale FANET can be deployed across future battlefields, providing operators on the ground with valuable real-time intelligence. Developing a FANET of nano drones requires solving complex problems regarding the trade-off of routing protocol security and resource constraints. This thesis offered a solution by integrating the lightweight cryptographic functions, Trivium and Chaskey-12 into the widely used AODV routing protocol.

This thesis designed and assessed the performance of the novel Lightweight Secured Ad Hoc On-Demand Distance Vector routing protocol (LS-AODV) in a FANET architecture. The performance evaluation of Trivium and Chaskey-12 suggested that using LS-AODV in a hardware environment will ensure a significant portion of power reserves can be directed towards other drone applications. The security evaluation of Trivium and Chaskey-12 highlighted the strength of LS-AODV to thwart a multitude of routing attacks. This thesis demonstrated that the LS-AODV routing protocol successfully maintains network connectivity while under attack from black-hole nodes.

In the non-threat FANET scenario, LS-AODV generally outperformed OLSR when analyzing PDR and routing overhead. However, because OLSR uses proactive routing techniques, it consistently induced less network jitter and delay than LS-AODV. AODV induced less network jitter and delay and had less routing overhead than LS-AODV because it does not have a cryptographic footprint. However, LS-AODV provided a higher PDR when the network contained more than 50 nodes and, for the most part, delivered more consistent and less fluctuating performance metrics. In the threat FANET scenario, LS-AODV steadily provided a much higher PDR and throughput than AODV and, in some instances, generated

less network delay and jitter.

6.2 Recommendations for Future Work

For future work, the following recommendations are provided:

- Implement the LS-AODV routing protocol in a hardware environment. Since Trivium and Chaskey-12 are primarily designed for hardware-based systems with limited resource capacity, this thesis indubitably underrepresents the potential of LS-AODV in real-world applications.

- Build a testbed comprised of fully functional PD-100 Black Hornet or equivalent nano drones. Currently, the PD-100 Black Hornet uses AES-256 encryption for video and photo transmission between a single drone and a mission control interface [6]. However, AES-256 may not offer the same benefits when the nano drones are incorporated into a densely populated swarm FANET, as simulated in this thesis.

- Study and test the Kreyvium [44] stream cipher as a replacement for Trivium in the LS-AODV routing protocol. Kreyvium is a 128-bit derivation of the Trivium stream cipher. However, with a larger key size comes a resource and latency tradeoff. Comparing Kreyvium and Trivium as encryption schemes within LS-AODV will provide greater insight into the delicate balance of routing security versus resource management.

- This thesis shows that LS-AODV performs well when operating under threat from a black-hole attack. However, other FANET routing attacks exist, such as the wormhole attack, rushing attack, and selfish node attack. An in-depth analysis of multiple attacks against LS-AODV may reveal which operations are essential to security and whether changes to the LS-AODV design paradigm might be necessary.

APPENDIX: Sample NS-3 Routing Protocol Comparison Program

This C++ script was used to build a FANET within NS-3 and test multiple routing protocols under various network conditions. The script provides several output mediums, including the command line, Flow Monitor parsed files, and the NetAnim interface.

```
#include <fstream>
#include <iostream>
#include "string"
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/mobility-module.h"
#include "ns3/aodv-module.h"
#include "ns3/olsr-module.h"
#include "ns3/config-store-module.h"
#include "ns3/netanim-module.h"
#include "ns3/LSAODV-module.h"
#include "ns3/wifi-module.h"
#include "ns3/applications-module.h"
#include "ns3/flow-monitor.h"
#include "ns3/flow-monitor-helper.h"
#include "ns3/flow-monitor-module.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("LS-AODV-Routing-Compare");

class RoutingExperiment
{
public:
```

```
  RoutingExperiment ();
  void Run (int nSinks, double txp, std::string CSVfile);
  std::string CommandSetup (int argc, char **argv);

private:
  Ptr<Socket> SetupPacketReceive(Ipv4Address addr,Ptr<Node>node);
  void ReceivePacket(Ptr<Socket>socket);
  void CheckThroughput();
  uint32_t port;
  uint32_t bytesTotal;
  uint32_t packetsReceived;
  std::string m_CSVfile;
  int m_nSinks;
  std::string m_protocolName;
  double m_txp;
  uint32_t m_protocol;
};

RoutingExperiment::RoutingExperiment()
  : port (9),
    bytesTotal (0),
    packetsReceived (0),
    m_CSVfile ("LS-AODV-Routing.Output.csv"),
    m_traceMobility (false),
    m_protocol (1)
{
}

static inline std::string
PrintReceivedPacket (Ptr<Socket> socket, Ptr<Packet> packet)
{
  SocketAddressTag tag;
  bool found;
```

```
  found = packet->PeekPacketTag (tag);
  std::ostringstream oss;

  oss << Simulator::Now ().GetSeconds () << " " <<
  socket->GetNode ()->GetId ();
  if (found)
    {
      InetSocketAddress addr =InetSocketAddress::
      ConvertFrom(tag.GetAddress ());
      oss << " received one packet from " << addr.GetIpv4 ();
    }
  else
    {
      oss << " received one packet!";
    }
  return oss.str ();
}
void RoutingExperiment::ReceivePacket (Ptr<Socket> socket)
{
  Ptr<Packet> packet;
  while ((packet = socket->Recv ()))
    {
      bytesTotal += packet->GetSize ();
      packetsReceived += 1;
      NS_LOG_UNCOND (PrintReceivedPacket (socket, packet));
    }
}
Ptr<Socket>
RoutingExperiment::SetupPacketReceive(Ipv4Address addr,
                                      Ptr<Node>node)
{
  TypeId tid = TypeId::LookupByName ("ns3::UdpSocketFactory");
  Ptr<Socket> sink = Socket::CreateSocket (node, tid);
```

```
  InetSocketAddress local = InetSocketAddress (addr, port);
  sink->Bind (local);
  sink->SetRecvCallback
  (MakeCallback (&RoutingExperiment::ReceivePacket,this));
  return sink;
}
std::string
RoutingExperiment::CommandSetup (int argc, char **argv)
{
  CommandLine cmd ;
  cmd.AddValue ("protocol","1=OLSR;2=AODV;3=LSAODV;...
  "4=MALICIOUS_AODV;5=MALICIOUS_LSAODV;",m_protocol);
  cmd.Parse (argc, argv);
  return m_CSVfile;
}

int main (int argc, char *argv[])
{
  RoutingExperiment experiment;
  std::string CSVfile = experiment.CommandSetup (argc,argv);
  std::ofstream out (CSVfile.c_str ());
  out << "SimulationSecond," <<
  "ReceiveRate," <<
  "PacketsReceived," <<
  "NumberOfSinks," <<
  "RoutingProtocol," <<
  "TransmissionPower" <<
  std::endl;
  out.close ();
  int nSinks = 10;
  double txp = 20;
  experiment.Run (nSinks, txp, CSVfile);
}
```

```
void RoutingExperiment::Run(int nSinks,double txp,std::
string CSVfile)
{
  Packet::EnablePrinting ();
  m_nSinks = nSinks;
  m_txp = txp;
  m_CSVfile = CSVfile;

  int nWifis = 20;

  double TotalTime = 200.0;
  std::string rate ("5120bps");
  std::string phyMode ("DsssRate11Mbps");
  std::string tr_name ("LSAODV-routing-compare");
  int nodeSpeed = 10; //in m/s
  int nodePause = 2; //in s
  m_protocolName = "protocol";

  uint32_t SentPackets = 0;
  uint32_t ReceivedPackets = 0;
  uint32_t LostPackets = 0;

  Config::SetDefault  ("ns3::OnOffApplication::PacketSize",
  StringValue ("512"));
  Config::SetDefault ("ns3::OnOffApplication::DataRate",
  StringValue (rate));
  Config::SetDefault ("ns3::WifiRemoteStationManager::
  NonUnicastMode", StringValue (phyMode));

  NodeContainer adhocNodes;
  NodeContainer not_malicious;
  NodeContainer malicious;
  adhocNodes.Create(nWifis);
```

```
WifiHelper wifi;
wifi.SetStandard (WIFI_PHY_STANDARD_80211b);

YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();
YansWifiChannelHelper wifiChannel;
wifiChannel.SetPropagationDelay (
"ns3::ConstantSpeedPropagationDelayModel");
wifiChannel.AddPropagationLoss ("ns3::FriisPropagationLossModel");
wifiPhy.SetChannel (wifiChannel.Create ());

WifiMacHelper wifiMac;
wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager",
                              "DataMode",StringValue(phyMode),
                              "ControlMode",StringValue(phyMode));

wifiPhy.Set ("TxPowerStart",DoubleValue (txp));
wifiPhy.Set ("TxPowerEnd", DoubleValue (txp));

wifiMac.SetType ("ns3::AdhocWifiMac");
NetDeviceContainer adhocDevices =
wifi.Install(wifiPhy,wifiMac,adhocNodes);

MobilityHelper mobilityAdhoc;
int64_t streamIndex = 0;

ObjectFactory pos;
pos.SetTypeId ("ns3::RandomRectanglePositionAllocator");
pos.Set ("X",StringValue(
"ns3::UniformRandomVariable[Min=0.0|Max=670.0]"));
pos.Set ("Y",StringValue(

Ptr<PositionAllocator> taPositionAlloc =
```

```
pos.Create ()->GetObject<PositionAllocator> ();
streamIndex += taPositionAlloc->AssignStreams(streamIndex);

std::stringstream ssSpeed;
ssSpeed<<"ns3::UniformRandomVariable[Min="
<<nodeSpeed<<"|Max="<<nodeSpeed<<"]";
std::stringstream ssPause;
ssPause<<"ns3::ConstantRandomVariable[Constant="<<nodePause <<"]";
mobilityAdhoc.SetMobilityModel("ns3::RandomWaypointMobilityModel",
              "Speed", StringValue (ssSpeed.str ()),
              "Pause", StringValue (ssPause.str ()),
              "PositionAllocator", PointerValue(taPositionAlloc));
mobilityAdhoc.SetPositionAllocator (taPositionAlloc);
mobilityAdhoc.Install (adhocNodes);
streamIndex += mobilityAdhoc.AssignStreams(adhocNodes,
streamIndex);

AodvHelper aodv;
AodvHelper malicious_aodv;
LSAODVHelper LSAODV;
LSAODVHelper malicious_LSAODV;
OlsrHelper olsr;
Ipv4ListRoutingHelper list;
InternetStackHelper internet;

switch (m_protocol)
  {
  case 1:
    list.Add (olsr, 100);
    m_protocolName = "OLSR";
    break;
  case 2:
    list.Add (aodv, 100);
```

```
    m_protocolName = "AODV";
    break;
  case 3:
    list.Add (LSAODV, 100);
    m_protocolName = "LSAODV";
    break;
  case 4:
    list.Add(malicious_aodv, 100);
    m_protocolName = "MALICIOUS_AODV";
    break;
  case 5:
    list.Add(malicious_LSAODV,100);
    m_protocolName = "MALICIOUS_LSAODV";
    break;
  default:
    NS_FATAL_ERROR ("No such protocol:" << m_protocol);
  }

if (m_protocol < 4)
  {
    internet.SetRoutingHelper (list);
    internet.Install (adhocNodes);
  }
else if(m_protocol == 4)
  {
    internet.SetRoutingHelper (aodv);
    internet.Install (not_malicious);
    malicious_aodv.Set("IsMalicious",BooleanValue(true));
    internet.SetRoutingHelper (malicious_aodv);
    internet.Install (malicious);
  }
else if(m_protocol == 5)
  {
```

```
    internet.SetRoutingHelper (LSAODV);
    internet.Install (not_malicious);
    malicious_LSAODV.Set("IsMalicious",BooleanValue(true));
    internet.SetRoutingHelper (malicious_LSAODV);
    internet.Install (malicious);
  }

NS_LOG_INFO ("assigning ip address");
Ipv4AddressHelper addressAdhoc;
addressAdhoc.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer adhocInterfaces;
adhocInterfaces = addressAdhoc.Assign (adhocDevices);
Ptr<OutputStreamWrapper> routingStream =
Create<OutputStreamWrapper> ("aodv.routes", std::ios::out);
aodv.PrintRoutingTableAllAt (Seconds (8), routingStream);

OnOffHelper onoff1 ("ns3::UdpSocketFactory",Address ());
onoff1.SetAttribute ("OnTime",
StringValue ("ns3::ConstantRandomVariable[Constant=1.0]"));
onoff1.SetAttribute ("OffTime",
StringValue ("ns3::ConstantRandomVariable[Constant=0.0]"));

for (int i = 0; i < nSinks; i++)
  {
    Ptr<Socket> sink = SetupPacketReceive (adhocInterfaces.
    GetAddress (i), adhocNodes.Get (i));
    AddressValue remoteAddress(InetSocketAddress(
    adhocInterfaces.GetAddress (i), port));
    onoff1.SetAttribute ("Remote", remoteAddress);
    Ptr<UniformRandomVariable> var =
    CreateObject<UniformRandomVariable> ();
    ApplicationContainer temp = onoff1.Install (adhocNodes.Get
    (i + nSinks));
```

```
    temp.Start (Seconds (var->GetValue (100.0,101.0)));
    temp.Stop (Seconds (TotalTime));
  }

std::stringstream ss;
ss << nWifis;
std::string sNodes = ss.str ();

std::stringstream ss2;
ss2 << nodeSpeed;
std::string sNodeSpeed = ss2.str ();

std::stringstream ss3;
ss3 << nodePause;
std::string sNodePause = ss3.str ();

std::stringstream ss4;
ss4 << rate;
std::string sRate = ss4.str ();

std::stringstream ss5;
ss5 << txp;
std::string sTxp = ss5.str ();

tr_name = m_protocolName + "_" + sNodes + "nodes";
m_CSVfile = tr_name;
FlowMonitorHelper flowmon;
Ptr<FlowMonitor> monitor = flowmon.InstallAll();

NS_LOG_INFO ("Run Simulation.");
Simulator::Stop (Seconds (TotalTime));
AnimationInterface anim (tr_name+"-animation.xml");
Simulator::Run ();
```

```
int j=0;
float AvgThroughput = 0;
Time Jitter;
Time Delay;

Ptr<Ipv4FlowClassifier>classifier = DynamicCast
<Ipv4FlowClassifier> (flowmon.GetClassifier ());
  std::map<FlowId, FlowMonitor::FlowStats> stats =
  monitor->GetFlowStats ();

for (std::map<FlowId, FlowMonitor::FlowStats>::
const_iterator iter =
stats.begin (); iter != stats.end (); ++iter)
  {
        Ipv4FlowClassifier::FiveTuple t =
        classifier->FindFlow (iter->first);
    NS_LOG_UNCOND("----Flow ID:" <<iter->first);
    NS_LOG_UNCOND("Src Addr" <<t.sourceAddress << "Dst Addr "<<
    t.destinationAddress);
    NS_LOG_UNCOND("Sent Packets=" <<iter->second.txPackets);
    NS_LOG_UNCOND("Received Packets =" <<iter->second.rxPackets);
    NS_LOG_UNCOND("Lost Packets =" <<
    iter->second.txPackets-iter->second.rxPackets);
    NS_LOG_UNCOND("Packet delivery ratio =" <<
    iter->second.rxPackets*100/iter->second.txPackets << "%");
    NS_LOG_UNCOND("Delay =" <<iter->second.delaySum);
    NS_LOG_UNCOND("Jitter =" <<iter->second.jitterSum);
    NS_LOG_UNCOND("Throughput =" <<iter->second.rxBytes * 8.0/
    (iter->second.timeLastRxPacket.GetSeconds()-
    iter->second.timeFirstTxPacket.GetSeconds())/1024/1024
    <<"Mbps");
```

```
      SentPackets = SentPackets +(iter->second.txPackets);
      ReceivedPackets = ReceivedPackets + (iter->second.rxPackets);
      LostPackets = LostPackets +
      (iter->second.txPackets-iter->second.rxPackets);
      AvgThroughput = AvgThroughput +(iter->second.rxBytes * 8.0/
      (iter->second.timeLastRxPacket.GetSeconds()-
      iter->second.timeFirstTxPacket.GetSeconds())/1024/1024);
      Delay = Delay + (iter->second.delaySum);
      Jitter = Jitter + (iter->second.jitterSum);
      j = j + 1;
   }
      AvgThroughput = AvgThroughput/j;
      NS_LOG_UNCOND("-Total Results of the simulation-"<<std::endl);
      NS_LOG_UNCOND("Total sent packets  =" << SentPackets);
      NS_LOG_UNCOND("Total Received Packets =" << ReceivedPackets);
      NS_LOG_UNCOND("Total Lost Packets =" << LostPackets);
      NS_LOG_UNCOND("Packet delivery ratio =" <<
      ((ReceivedPackets*100)/SentPackets)<< "%");
      NS_LOG_UNCOND("Average Throughput ="<<AvgThroughput<<"Mbps");
      NS_LOG_UNCOND("Average Network Delay =" << Delay/j);
      NS_LOG_UNCOND("Network Jitter =" << Jitter/j);
      NS_LOG_UNCOND("Total Flow id " << j);

   monitor->SerializeToXmlFile((tr_name +".flowmon").
   c_str(),true,true);
   Simulator::Destroy ();
}
```

List of References

[1] M. F. Khan, K.-L. A. Yau, R. M. Noor, and M. A. Imran, "Routing schemes in FANETs: A survey," *Sensors*, vol. 20, no. 1, p. 38, 2020 [Online]. Available: https://doi.org/10.3390/s20010038

[2] Department of Defense, *Unmanned Aircraft System Airspace Integration Plan, Version 2.0*, Washington, DC: USA, 2011 [Online]. Available: https://www.hsdl.org/?view&did=723337

[3] Naval Undersea Warfare Center Division Newport, "Evolving command, control and communications for unmanned systems," Jan. 8, 2021 [Online]. Available: https://www.dvidshub.net/news/386617/evolving-command-control-and-communications-unmanned-systems

[4] M. Karaduman, A. Çınar, and H. Eren, "UAV traffic patrolling via road detection and tracking in anonymous aerial video frames," *Journal of Intelligent Robotic Systems*, vol. 95, no. 2, pp. 675–690, 2019 [Online]. Available: http://dx.doi.org/10.1007/s10846-018-0954-x

[5] J. Sanborn, "Marines get a closer look at Black Hornet micro drone," Marine Corps Times, Sep. 22, 2015 [Online]. Available: https://www.marinecorpstimes.com/news/your-marine-corps/2015/09/22/marines-get-a-closer-look-at-black-hornet-micro-drone/

[6] Teledyne FLIR, "Black Hornet PRS Airborne Personal Reconnaissance System (PRS)," Accessed Feb. 10, 2022 [Online]. Available: https://www.flir.com/products/black-hornet-prs/

[7] C. Perkins, "Ad hoc On-Demand Distance Vector (AODV) Routing," RFC 3561, 2003 [Online]. Available: https://www.ietf.org/rfc/rfc3561.txt

[8] C. Perkins, *Ad Hoc Networking*, 2nd ed. Addison Wesley, 2004.

[9] C. De Canniere and B. Preneel, "Trivium: A stream cipher construction inspired by block cipher design principles," in *Information Security*. Berlin, Heidelberg: Springer, 2006 [Online]. Available: https://doi.org/10.1007/11836810_13

[10] NSNAM, "Ns-3 Model Library," Accessed Feb. 01, 2022 [Online]. Available: https://www.nsnam.org/docs/release/3.25/models/singlehtml/index.html

[11] NSNAM, "Ns-3 Manual," Accessed Feb. 01, 2022 [Online]. Available: https://www.nsnam.org/docs/release/3.25/manual/singlehtml/

[12] A. Bujari, C. T. Calafate, J. C. Cano, P. Manzoni, C. E. Palazzi, and D. Ronzani, "Flying ad-hoc network application scenarios and mobility models," *International Journal of Distributed Sensor Networks*, vol. 13, no. 10, 2017 [Online]. Available: https://doi.org/10.1177/1550147717738192

[13] J. Li, Y. Zhou, L. Lamont, M. Toulgoat, and C. A. Rabbath, "Packet delay in UAV wireless networks under non-saturated traffic and channel fading conditions," *Wireless Personal Communications*, vol. 72, no. 2, pp. 1105–1123, 2013 [Online]. Available: https://doi.org/10.1007/s11277-013-1057-4

[14] M. B. Yassein and N. A. Damer, "Flying ad-hoc networks: Routing protocols, mobility models, issues," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 7, no. 6, 2016 [Online]. Available: https://dx.doi.org/10.14569/IJACSA.2016.070621

[15] A. Guillen-Perez, A.-M. Montoya, J.-C. Sanchez-Aarnoutse, and M.-D. Cano, "A comparative performance evaluation of routing protocols for flying ad-hoc networks in real conditions," *Applied Sciences*, vol. 11, no. 10, p. 4363, 2021 [Online]. Available: https://doi.org/10.3390/app11104363

[16] M. T. Hyland, "Performance evaluation of ad hoc routing protocols in a swarm of autonomous unmanned aerial vehicles," M.S. thesis, Department of the Air Force Air University, Wright-Patterson Air Force Base, Ohio, USA, 2007 [Online]. Available: https://apps.dtic.mil/sti/citations/ADA469161

[17] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol (OLSR)," RFC 3626, 2003 [Online]. Available: https://www.ietf.org/rfc/rfc3626.txt

[18] U.S. Naval Research Laboratory, "Optimized Link State Routing (OLSR)," Accessed Feb. 01, 2022 [Online]. Available: https://www.nrl.navy.mil/Our-Work/Areas-of-Research/Information-Technology/NCS/OLSR/)/

[19] H. Deng, W. Li, and D. Agrawal, "Routing security in wireless ad hoc networks," *IEEE communications magazine*, vol. 40, no. 10, pp. 70–75, 2002 [Online]. Available: https://doi.org/10.1109/MCOM.2002.1039859

[20] M. Zapata and N. Asokan, "Securing ad hoc routing protocols," in *Proceedings of the 1st ACM Workshop on Wireless Security*, 2002, pp. 1–10.

[21] C. Ran, S. Yan, L. Huang, and L. Zhang, "An improved AODV routing security algorithm based on blockchain technology in ad hoc network," *EURASIP journal on wireless communications and networking*, vol. 2021, no. 1, pp. 1–16, 2021.

[22] D. R. Choudhury, L. Ragha, and N. Marathe, "Implementing and improving the performance of AODV by receive reply method and securing it from black hole attack," *Procedia Computer Science*, vol. 45, pp. 564–570, 2015 [Online]. Available: https://doi.org/10.1016/j.procs.2015.03.109

[23] N. W. Aziz, S. N. Alsaad, and H. K. Hmood, "Implementation of lightweight stream cipher in AODV routing protocol for MANET," in *2019 First International Conference of Computer and Applied Sciences (CAS)*, 2019 [Online]. Available: https://ieeexplore.ieee.org/document/9075669

[24] M. Dworkin, E. Barker, J. Nechvatal, J. Foti, L. Bassham, E. Roback, and J. Dray, *Advanced Encryption Standard (AES)*, Federal Inf. Process. Stds. (NIST FIPS), National Institute of Standards and Technology, Gaithersburg, MD, 2001 [Online]. Available: https://doi.org/10.6028/NIST.FIPS.197

[25] C. De Canniére and B. Preneel, "Trivium specifications," eSTREAM, ECRYPT Stream Cipher Project, Report 2005/030, 2005 [Online]. Available: https://www.ecrypt.eu.org/stream/p3ciphers/trivium/trivium_p3.pdf

[26] *Information technology — Security techniques — Lightweight cryptography — Part 3: Stream ciphers*, ISO/IEC Standard 29192-3, 2012 [Online]. Available: https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/05/64/56426.html

[27] L. R. Knudsen and M. J. B. Robshaw, "AES," in *The Block Cipher Companion*. Berlin, Heidelberg: Springer, 2011 [Online]. Available: https://doi.org/10.1007/978-3-642-17342-4_3

[28] M. Feldhofer, "Comparison of low-power implementations of trivium and grain," eSTREAM, ECRYPT Stream Cipher Project, Report 2007/027, 2007 [Online]. Available: https://www.ecrypt.eu.org/stream/papersdir/2007/027.pdf

[29] S. Banik, "Towards low energy stream ciphers," *IACR Transactions on Symmetric Cryptology*, pp. 1–19, 2018 [Online]. Available: https://doi.org/10.13154/tosc.v2018.i2.1-19

[30] T. Good and M. Benaissa, "ASIC hardware performance," in *New Stream Cipher Designs: The eSTREAM Finalists*. Berlin, Heidelberg: Springer, 2008 [Online]. Available: https://doi.org/10.1007/978-3-540-68351-3_19

[31] A. Maximov and A. Biryukov, "Two trivial attacks on Trivium," in *Selected Areas in Cryptography*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 36–55.

[32] F. E. Potestad-Ordonez, M. Valencia-Barrero, C. Baena-Oliva, P. Parra-Fernandez, and C. J. Jimenez-Fernandez, "Breaking Trivium stream cipher implemented in ASIC using experimental attacks and DFA," *Sensors*, vol. 20, no. 23, 2020 [Online]. Available: https://doi.org/10.3390/s20236909

[33] N. Mouha, B. Mennink, A. Van Herrewege, D. Watanabe, B. Preneel, and I. Verbauwhede, "Chaskey: An efficient MAC algorithm for 32-bit microcontrollers," in *Selected Areas in Cryptography – SAC 2014*, 2014, pp. 306–323.

[34] J.-P. Aumasson and D. J. Bernstein, "SipHash: A fast short-input PRF," in *Progress in Cryptology - INDOCRYPT 2012*, 2012 [Online]. Available: https://doi.org/10.1007/978-3-642-34931-7_28

[35] *Information technology — Lightweight cryptography — Part 6: Message authentication codes (MACs)*, ISO/IEC Standard 29192-6, 2019 [Online]. Available: https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/07/11/71116.html

[36] CryptoLUX, "FELICS block ciphers brief results," Accessed Feb. 01, 2022 [Online]. Available: https://www.cryptolux.org/index.php/FELICS_Block_Ciphers_Brief_Results

[37] C. Patrick and P. Schaumont, "The role of energy in the lightweight cryptographic profile," unpublished.

[38] S. Even and Y. Mansour, "A construction of a cipher from a single pseudorandom permutation," in *Advances in Cryptology — ASIACRYPT '91*. Berlin, Heidelberg: Springer, 1993 [Online]. Available: https://doi.org/10.1007/3-540-57332-1_17

[39] J. Black, S. Halevi, H. Krawczyk, T. Krovetz, and P. Rogaway, "UMAC: Fast and secure message authentication," in *Advances in Cryptology — CRYPTO' 99*, 1999, pp. 216–233.

[40] A. D. Dwivedi, "Security analysis of lightweight IoT cipher: Chaskey," in *Cryptography*, 2020 [Online]. Available: https://www.mdpi.com/2410-387X/4/3/22

[41] S. Corson, "Mobile Ad hoc Networking (MANET): Routing protocol performance issues and evaluation considerations," RFC 2501, 1999 [Online]. Available: https://www.ietf.org/rfc/rfc2501.txt

[42] T. Clausen, "Jitter considerations in Mobile Ad Hoc Networks (MANETs)," RFC 5148, 2008 [Online]. Available: https://www.ietf.org/rfc/rfc5148.txt

[43] C. M. Demichelis and P. Chimento, “IP packet delay variation metric for IP performance metrics (IPPM),” RFC 3393, 2002 [Online]. Available: https://www.rfc-editor.org/info/rfc3393

[44] A. Canteaut, S. Carpov, C. Fontaine, T. Lepoint, M. Naya-Plasencia, P. Paillier, and R. Sirdey, “Stream ciphers: A practical solution for efficient homomorphic-ciphertext compression,” in *Fast Software Encryption*, 2016, pp. 313–333.

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California