

BOTSWANA INTERNATIONAL UNIVERSITY
OF SCIENCE & TECHNOLOGY

BURST CONTROL PACKET SECURITY IN OBS NETWORKS

BY

MPHO DAVID

ID: 13000461

MEng COMPUTER AND TELECOMMUNICATION ENGINEERING
A THESIS SUBMITTED TO THE FACULTY OF ENGINEERING IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF MASTER OF COMPUTER AND TELECOMMUNICATION ENGINEERING MASTER OF ENGINEERING

JUNE 2021

Burst Control Packet Security in OBS Networks

By
Mpho DAVID
Reg No : 13000461

Department of Electrical, Computer and Telecommunication Engineering
Faculty of Engineering and Technology
Botswana International University of Science and Technology
mpho.david@studentmail.biust.ac.bw

A thesis Submitted to the Faculty of Engineering In Partial fulfilment of the Requirements for the Award of the Degree of Master of Computer and Telecommunication Engineering Master of Engineering

Supervisor(s):
Dr. Mmoloki Mangwala
Department of Electrical, Computer and Telecommunications
Faculty of Engineering, BIUST
mangwalam@biust.ac.bw
Signature:
_____

Date:
_____

Dr. Boyce Sigweni
Department of Electrical, Computer and Telecommunications
Faculty of Engineering, BIUST
sigwenib@biust.ac.bw
Signature:
_____

Date:
_____

June 18, 2021

Declaration of Authorship

I, Mpho DAVID, declare that this thesis titled, "Burst Control Packet Security in OBS Networks" and the work presented in it are my own, is my original work, which has never been or will never be submitted to any other university for a similar or other degree award.

Signed: M. David

Date: 23/06/2021

This dissertation/thesis is copyright material protected under the Berne Convention, the Copyright Act of 1999 and other international and national enactments, in that behalf, on intellectual property. It must not be reproduced by any means, in full or in part, except for short extracts in fair dealing; for researcher private study, critical scholarly review or discourse with an acknowledgement, without the written permission of the office of the DVCAA, on behalf of both the author and the BIUST.

Certification

The undersigned certifies that he/she has read and hereby recommends for acceptance by the Faculty of Engineering a dissertation/thesis titled: Burst Control Packet Security in OBS Networks, in fulfilment of the requirements for the degree of Master of Engineering in (Computer and Telecommunications) of the BIUST.

ACTING HOD

for **Dr. Mmoloki Mangwala**

23/06/2021

Date

Dr. Boyce Sigweni

Date

Acknowledgements

Sincere gratitude is expressed to my supervisors Dr. Mmoloki Mangwala and Dr. Boyce Sigweni, who have been my mentors throughout my research work. The motivation, guidance, and unending patience they gave has led to the successful completion of this thesis. I thanked them for the consistent and reliable support even when their schedules were full or times of exhaustion.

Dr. Mafuraga has my undying appreciation for his invaluable assistance, which has had a huge influence on the literary progress of my thesis. I'd also like to express my gratitude to the Botswana International University of Science and Technology's staff and students for their encouragement and support during the research process.

This thesis is dedicated to my family for the support they have given to me throughout my life. They have molded me into the man I am today and I will forever be grateful to them.

Abstract

Optical networks gained interest due to the inadequacy of electrical networks to handle the rapid growth of bandwidth requirements, as optical networks have significantly higher transmission rates. To efficiently realize optical networks three standards namely; optical packet switching, optical burst switching (OBS) and optical packet switching, were developed o take advantage of wavelength division multiplexing to implement optical communication. In OBS the control packet undergoes Optical-Electrical-Optical (O-E-O) conversion at every intermediate core node. The problem is, if a node is compromised it can subsequently act as a gateway to the entire network and an attacker can have access to the control packet as it undergoes processing at that node. This attacker can then modify any field in the control packet and hence, introduce a number of security concerns by compromising the integrity, confidentiality and integrity of the control packet and by default that of the data burst. This work therefore proposes the development and testing of a secure burst control packet mechanism that uses SHA-256 for per hop authentication to ensure the integrity of the control packet and AES to ensure the confidentiality of the control packet. In the development of the proposed mechanism, experiments showed it was able secure the control packet as it travels throughout the network and maximized network resource utilization without incurring more end-to-end delay. Additionally, the proposed algorithm is faster than asymmetric cryptographic techniques however is slower than symmetric cryptographic techniques. The proposed algorithm can be used instead of asymmetric algorithms as it provides higher throughput and lower end-to-end delay. The proposed algorithm provides maximum resource utilization as such can be used instead of the nodal isolation techniques.

Contents

Declaration of Authorship i

Acknowledgements i

Abstract iii

List of Figures vii

List of Tables ix

List of Abbreviations x

Mathematical Notations xi

1 Introduction 1

1.1 Introduction 1

1.2 Background 3

1.2.1 Optical Burst Switching 3

1.2.2 Challenges in OBS 4

1.2.2.1 Burst assembly 4

1.2.2.2 Contention 5

1.2.2.3 Signalling 5

1.2.2.4 Scheduling 6

1.2.2.5 Security 6

1.3 Problem statement 6

1.4 AIM 7

1.5 Specific Objectives 7

1.6 Research Questions 7

1.7 Expected Outcome 8

1.8 Thesis Outline 8

1.8.1 Chapter 2: Literature review 8

1.8.2 Chapter 3:Methodology 8

1.8.3 Chapter 4:Results Analysis 8

Contents

Declaration of Authorship i

Acknowledgements i

Abstract iii

List of Figures vii

List of Tables ix

List of Abbreviations x

Mathematical Notations xi

1 Introduction 1

1.1 Introduction 1

1.2 Background 3

1.2.1 Optical Burst Switching 3

1.2.2 Challenges in OBS 4

1.2.2.1 Burst assembly 4

1.2.2.2 Contention 5

1.2.2.3 Signalling 5

1.2.2.4 Scheduling 6

1.2.2.5 Security 6

1.3 Problem statement 6

1.4 AIM 7

1.5 Specific Objectives 7

1.6 Research Questions 7

1.7 Expected Outcome 8

1.8 Thesis Outline 8

1.8.1 Chapter 2: Literature review 8

1.8.2 Chapter 3:Methodology 8

1.8.3 Chapter 4:Results Analysis 8

1.8.4 Chapter 5:Conclusion 8

2 Literature Review **9**
2.1 Control Packet Structure 9
2.2 Security Threats in OBS 10
2.3 Security Threat Detection Methods 12
2.3.1 Evaluation metrics of threat detection methods in Optical Burst Switched networks 13
2.3.1.1 Simulation 13
2.3.1.2 Confusion Matrix 14
2.4 Security Threat Handling Methods in OBS 14
2.4.1 Evaluation Metrics Used to Evaluate Threat Handling Mechanisms in OBS Networks 16
2.5 Conclusion 17

3 Methodology **19**
3.1 Introduction 19
3.2 Control Packet Authentication 19
3.2.1 Hash functions 20
3.2.1.1 MD4 23
3.2.1.2 MD5 25
3.2.1.3 SHA-1 26
3.2.1.4 SHA-256 29
3.2.1.5 RIPEMD-160 30
3.2.2 Speed and Security of Hash Functions 31
3.3 Security Protocol Method Selection 33
3.3.1 Symmetric Algorithms 34
3.3.1.1 Data Encryption Standard (DES) 34
3.3.1.2 Triple Data Encryption Standard (3DES) 38
3.3.1.3 The Advanced Encryption Standard (AES) 39
3.3.1.4 Blowfish 48
3.3.2 Security of Symmetric Algorithms 49
3.3.3 Speed of Symmetric Algorithms 50
3.4 Algorithm Formulation 51
3.5 Conclusion 57

4 Results Analysis **58**
4.1 Introduction 58
4.2 Evaluation Framework 58

4.3 Simulation Scenarios 62
4.4 Performance Metrics 64
4.4.1 End-to-end delay 64
4.4.2 Throughput 65
4.4.3 Burst Loss Ratio 65
4.5 Experimental Results 66
4.5.1 Throughput 66
4.5.2 Burst Loss Ratio 68
4.5.3 End-to-End Delay 70
4.6 Summary 72

5 Conclusion **73**
5.1 Conclusion 73
5.2 Summation of Accomplished Work 73

Appendix A **76**

Appendix B **78**

Appendix C **81**

Bibliography **84**

List of Figures

1.1 Development of WDM optical networks [7] 2
1.2 Architecture of OBS network 3

2.1 control packet structure 9

3.1 View of the Merkel Damgard construction reproduced from [91] . . . 22
3.2 Input and output parameters of DES obtained from [27] 35
3.3 feistel structure of DES reprinted from [103] 36
3.4 Triple DES structure obtained from [27] 39
3.5 Input output parameters of AES reprinted from [27] 39
3.6 encryption structure of AES reprinted from [104] 40
3.7 S-box mathematical operations 44
3.8 structure of the Blowfish algorithm 49

4.1 Edge node architecture 60
4.2 core node architecture 61
4.3 simulation topology 62
4.4 simulation scenario 63
4.5 received bursts with varying traffic for normal scenario 66
4.6 throughput when 5 nodes are compromised 67
4.7 throughput when 10 nodes are compromised 68
4.8 burst loss ratio for normal scenario 68
4.9 burst loss ratio when 5 nodes are compromised 69
4.10 burst loss ratio when 10 nodes are compromised 69
4.11 end-end delay normal scenario 70
4.12 end to end delay when 5 nodes are compromised 71
4.13 end to end delay when 10 nodes are compromised 71

A.1 compression function of sha-1 76
A.2 compression function of MD5 algorithm 76
A.3 compression function of sha256 algorithm 77
A.4 compression function of RIPEMD-160 algorithm 77

C.1 Round coefficient of 128 bit key schedule 82
C.2 S-box function in 256-bit key schedule 83

List of Tables

2.1 Identification methods 12
2.2 simulators used for detection 14
2.3 Features of threat handling methods 16

3.1 First Permutation 37
3.2 Final Permutation 38
3.3 AES S-Box: Hexadecimal representation of substitution values obtained from [27] 44
3.4 Different encryption times(in milliseconds) of symmetric algorithms with varying packet size 50
3.5 Different decryption times(in milliseconds) of symmetric algorithms with varying packet size 51

4.1 simulation parameters 62

5.1 Exisitng OBS security mechanisms 74
5.2 Comparison of simulated algorithm's based on throughput, end-to-end delay and burst loss ratio 75

B.1 first substitution box 78
B.2 second substitution box of DES algorithm 78
B.3 Third substitution box of DES algorithm 78
B.4 Fourth substitution box of DES algorithm 78
B.5 fifth substitution box of DES algorithm 79
B.6 Sixth substitution box of DES algorithm 79
B.7 Seventh substitution box of DES algorithm 79
B.8 Eighth substitution box of DES algorithm 79
B.9 lookup table for Expansion permutation in DES 79
B.10 Permutation table in DES 80

C.1 Inverse multiplicative table used within the AES S-Box obtained from [27] 81

List of Abbreviations

3DES	Triple Data Encryption Standard
λID	Wavelength Channel Identifier
AES	Advanced Encryption Standard
CRC	Cyclic Redundancy Check
DES	Data Encryption Standard
GF	Galois Fields
IP	Internet Protocol
MD	Merkel Damgard
MD4	Message digest 4
MD5	Message digest 5
MPLS	Multi Protocol Label Switching
NSFNET	National Science Foundation Network
OBS	Optical Burst Switching
OCS	Optical Circuit Switching
OEO	Optical-Electronic-Optical
OMNET	Objective Modular Network Testbed
OPS	Optical Packet Switching
OXC	Optical Cross Connect
RIPEMD	Race Integrity Primitives Evaluation Message Digest
RSA	Rivest-Shamir-Adleman
SCU	Switching Control Unit
SHA	Secure Haashing Algorithm
SI	Segment Indicator
SL	Segment Length
QOS	Quality of Service
WADM	Wavelength Add and Dropp Multiplexing
WDM	Wavelength Division Multiplexing

Mathematical Notations

$\wedge$	The logical conjunction operator.	The and of a set of operands is true if and only if all of its operands are true.
$\vee$	The logical disjunction operator	for a set of operands is true if and only if at least one of them is true.
$\oplus$	exclusive disjunction	exclusive disjunction is a logical operation that is true if and only if its arguments differ.
$\neg$	Negation	Negation, often known as the logical complement, is a logic operation that converts one proposition to another.
$\ll$	Bitwise left shift operator	This operator shifts bits to the left.
$\gg$	Bitwise right shift operator	This operator shifts bits to the right.
$\ggg$	Right rotate	Right rotate cyclically shift bits to the right.
$\mathbb{Z}$	integer ring	An integer ring is an abelian group containing the addition operation and a second binary operation, multiplication, that is associative, distributive over addition, and possesses a multiplicative identity element.

Chapter 1

Introduction

1.1 Introduction

Over the last decade, bandwidth requirements have increased significantly every year [1], [2]. Unfortunately, infrastructure upgrades have not been commensurate with the rise in demand [3]. Predictably the consequence of this demand supply mismatch has led to significant strain on existing telecommunications infrastructure, thus pushing for the development and widespread adoption of optical fibre networks as they provide transmission at higher rates and capacity [4], [5]. This is made possible by Wavelength Division Multiplexing (WDM) technology [6] where multiple carrier signals operating on different wavelengths are multiplexed together into a single optical fibre allowing a single fibre to transmit with speeds eclipsing 50 Tb/s. The evolution of optical network transport techniques is classified as shown in Figure 1.1.

Point-to-point WDM networks consist of links that process data electronically, optical to electrical conversion needs to occur for processing, the data needs to be converted back to optical for re-transmission [8]. During O-E-O conversion huge overheads are incurred due to the electronic processing cost and switching complexity [9], especially if most of the data is bypass traffic. To address some of the short falls in point-to-point networks, Wavelength Add/Drop Multiplexer networks were developed [10]. In WADM networks some channels can be terminated while others are allowed to pass through without undergoing optical electrical optical (O-E-O) conversion [11]. This means that bypass traffic can pass through the node unaffected hence reducing network cost. WADM networks are mostly used in ring networks. The development of interconnection devices which are used to transfer data in a network made it possible to build mesh networks [12]. These devices can be classified as Active Switches [13], Passive Star Couplers [14] and Passive Routers [15]. These devices made it possible to produce mesh networks and three major switching

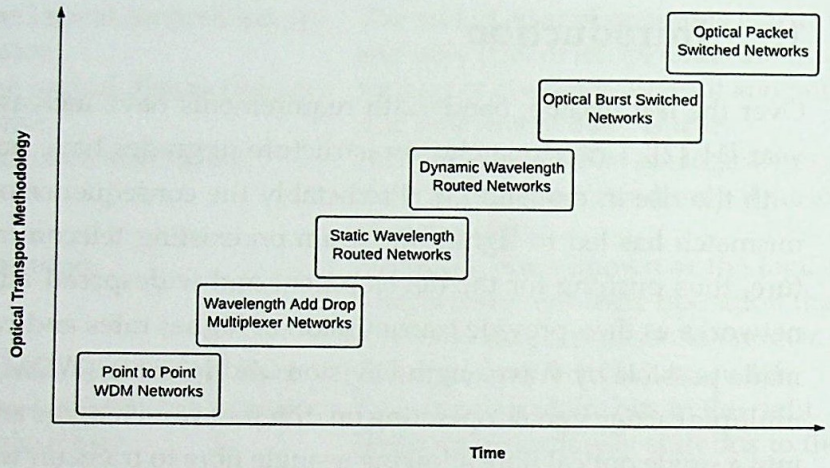


FIGURE 1.1: Development of WDM optical networks [7]

transport methodologies for transporting TCP/IP over WDM [16] emerged. These methodologies are:

- Optical Circuit Switching (OCS) where a connection is established on a predefined wavelength from source to destination before transmission commences [17], [18]. Light paths can be connected dynamically or statically depending on traffic hence can be referred to as dynamic wavelength routed networks or static wavelength routed networks. OCS incurs high overhead as it has high setup latency, low bandwidth utilization, low flexibility and hence cannot handle the bursty nature of internet traffic [19].

- Optical Packet Switching (OPS) where each packet is switched individually in the network with its control information carried on its corresponding control packet [20], [21]. Resources and technology to implement OPS are still in their infancy stage so OPS is not feasible in the immediate future [2].

- Optical Burst Switching (OBS) where packets heading to the same destination are grouped to produce a data burst and switched together in the network [22], [3]. A control packet carrying control information for the burst is sent ahead to

reserve network resources. OBS is a compromise between OCS and OPS such that it implements all the benefits of OPS while at the same time avoiding all the limitations faced by OCS [22].

1.2 Background

1.2.1 Optical Burst Switching

Optical Burst Switching (OBS) is intended to find some kind of harmony between OCS and OPS. In OBS, information bursts comprising of numerous IP packets are exchanged optically through the network. A data burst is a packet of varying length, assembled by aggregating multiple IP data packets at the ingress node [23], [24]. The control packet, which is sent before the burst, configures the switches. The control packet consists of: burst length, the destination address of the burst and burst source address. The offset time is the time difference between the burst and the control packet which is used to configure switches hence no buffering of the burst is required in the network [25], [24]. The offset time is set such that it is equal or greater than the processing delay on the control packet at a core node during processing. The burst length carried in the burst header packet is used to release resources once the burst has been transmitted through the node.

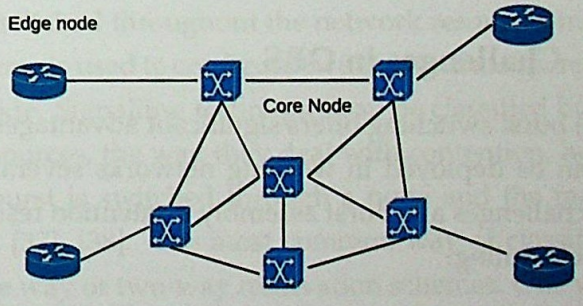


FIGURE 1.2: Architecture of OBS network

As shown in Figure 1.2, OBS networks are made up of optical burst switched nodes linked by an optical fibre. OBS networks consists of two nodes being Edge and Core nodes:

- Edge nodes

Edge nodes link the OBS network and the client network. They consist of electronic buffers that are used to store IP packets before sorting them into bursts and OBS interface. Ingress nodes and egress nodes are the two types of edge nodes [26]. Edge node have six main functions being; burst assembly, control packet generation, routing and wavelength assignment, offset computation, burst disassembly and signalling. Of the six mentioned functions the ingress node carries out the following duties:

1. Burst assembly
2. Control packet generation
3. Wavelength assignment and routing
4. Scheduling of bursts at the ingress node

The duties of the egress node include; burst disassembly and the dispatching of packets to network layers above. [26].

- Core nodes

Core nodes comprises of a Switching Control Unit (SCU) and an optical cross connect (OXC) [27]. The SCU is used to configure the OXC based on the control information provided by the control packet.

1.2.2 Challenges in OBS

Optical burst switching offers significant advantages over OCS and OPS but before OBS can be deployed in working networks several issues need to be addressed. These challenges are; burst assembly, contention resolution, security, signalling and burst scheduling.

1.2.2.1 Burst assembly

For packets to be assembled into bursts a burst assembly scheme is required. This scheme should take into account when to aggregate packets,the quantity of IP packets a burst should have and the type of packets a burst must posses [28], [29]. The most commonly used schemes are the threshold based and timer based schemes [30], [31]. A timer scheme aggregates bursts based on a timer set once the aggregation process commences [28]. This scheme provides bursts at uniform intervals into the network. If a large time out value is set and contention occurs, then a high volume of bursts will be lost per contention. However, if a small time out value is set then there will be a lot of bursts in the network which in turn increases the chances

of contention occurring.

A threshold scheme sets a ceiling on a the number of packets required in a single burst. Once the number is attained the burst is sent into the network. If a large threshold is set, then a high volume of packets will be lost per contention. However, if a small threshold is set then there will a lot of bursts in the network increasing the chances of contention [32].

1.2.2.2 Contention

When more than one burst requests to use the same output port at the same time, contention occurs [32]. Unlike in the electrical domain where buffers are usually used to deal with contention, it is challenging to have buffers in the optical domain, therefore other measures are taken to deal with contention [33], [34]. These include [35], [36]:

- Wavelength conversion changes the current wavelength of the burst into a free wavelength on the output port.
- Deflection routing
- Burst segmentation
- Fiber delay lines

1.2.2.3 Signalling

Before a burst can be switched throughout the network resources must be reserved for it. A signalling scheme is used to configure switches and assign resources needed by the burst at each node. Signalling techniques can be classified by the manner in which they reserve resources, the way they deal with contention, how they release the resources after a burst is switched through a node and the manner in which they reserve resources [37], [38]. The most common way of classifying signalling techniques is either one way or two-way reservation schemes. One way reservation sends a burst without waiting for an acknowledgement while two way reservations is acknowledgement based [39]. The most widely used signalling techniques are Tell and Go where a burst is transmitted instantly after the control packet without knowing if the switches have been configured for it. Another signalling technique mostly used is the tell and wait where a burst waits for an acknowledgement from the node before it is injected into the network [35].

1.2.2.4 Scheduling

At every core node in the network bursts must be assigned to a pertinent output port wavelength. In every output link there exists a gap between two scheduled burst. Scheduling schemes are primarily focused on minimizing this gap [40], [41]. There are two classes of scheduling algorithms; with void filling and without void filling or Horizon [42]. Void filling algorithms take into account the voids between bursts when scheduling. Without void scheduling algorithms do not take into account the void between bursts when scheduling [43].

1.2.2.5 Security

The control packet is processed by every core node that receives it. If a core node is compromised in a network an attacker can compromise the control packet leading to a number of security vulnerabilities [44]. Security threats in OBS can be categorized into two [45]:

- Orphan burst

This results when a scheduling request is denied at a node due to contention. The burst will arrive at this node and will not be routed as no reservation is made for it. This burst is referred to as an Orphan burst. This burst will flow in the network wasting bandwidth and will end up at an unintended egress node [46].

- Malicious Burst header

If a node is compromised the control packet fields can be altered as such compromise the confidentiality, integrity and availability of data [45].

1.3 Problem statement

The exponential growth of the quantity of users in the internet and the increase in bandwidth required by applications has made optical networks all the more relevant, with optical burst switching being the most ideal as it has the capability of transferring data at tremendous speeds. However, the control packet is plagued by a variety of security challenges as it traverses the network. A control packet undergoes O-E-O conversion at every intermediate core node. If a node is compromised it can subsequently act as a gateway to the entire network and an attacker can have access to the control packet as it undergoes processing at that node. This attacker can then modify any field in the control packet and hence, introduce a number of security concerns by compromising the integrity, confidentiality and integrity of the control packet and by default that of the data burst.

1.4 AIM

Optical Burst Switching has the capability to drastically improve the provisioning of end-to-end light path establishment between communicating entities in a network. However, security threats in OBS jeopardize the integrity, confidentiality and authenticity of data. Most of these security threats are aimed towards the control packet; hence the development of a mechanism that enhances the security of the control packet will drastically improve the security of data throughout the network. The aim of this research is to investigate different security concerns relating to a control packet of a data burst along with their proposed solutions and to design a secure burst control packet mechanism to counteract identified security threats.

1.5 Specific Objectives

1. To review all existing literature pertaining to security concerns in optical burst switching in relation to the control packet how the threats are detected and ways of mitigating these threats.
2. To formulate a secure burst control packet mechanism for optical burst switching network based on information obtained on the above objective
3. To evaluate the proposed mechanism by comparing results to existing mechanisms

1.6 Research Questions

The purpose of this study is to find answers to the following inquiries:

1. What are the major security concerns in optical burst switched networks related to control packets?
2. How are security threats identified in OBS networks?
3. What are the security mechanisms dealing with security concerns related to control packets in OBS networks?
4. What are the trade-offs introduced by the mechanisms identified?
5. On which metrics are the mechanisms dealing with security concerns of control packets in OBS networks gauged?

1.7 Expected Outcome

The expected outcome of this study is to design a secure mechanism for the control packet that secures it as it travels through the core nodes of the network. This mechanism will ensure the confidentiality of data through the network as it ensures that only the intended recipients have access to the data.

1.8 Thesis Outline

The thesis is organized in five chapters which are further broken down into their related sub-sections. A breakdown of the thesis is given below.

1.8.1 Chapter 2: Literature review

The second chapter discusses emerging processes in order to obtain a deeper understanding of threat detection strategies currently in use, how security vulnerabilities are handled and weaknesses of existing security mechanisms. An in-depth look into the structure of the control packet and its associated security vulnerabilities.

1.8.2 Chapter 3:Methodology

The third chapter describes a framework that is to be followed to design and simulate a secure burst control packet mechanism for OBS networks. The steps are clearly outlined to provide a better understanding of the work undertaken.

1.8.3 Chapter 4:Results Analysis

This chapter discusses the simulation results obtained after testing the mechanism.

1.8.4 Chapter 5:Conclusion

The final chapter evaluates if the proposed methodology addressed the objectives outlined. A comprehensive validation of the methods used is provided in this chapter.

Chapter 2

Literature Review

This section discusses the structure of the control packet, security vulnerabilities in OBS associated with the control packet, how these threats are detected and how these threats are handled in the network. This is all achieved by exploring existing literature.

2.1 Control Packet Structure

Every data burst has a corresponding control packet that is sent ahead of it to configure and reserve resources needed by the burst. The control packet format allows for continuous transmission overhead. Its use of the segment indicator instead of flags ensures that the control packet is always scalable to higher transmission speeds [47]. Figure 2.1 provides the structure of the control packet and a brief explanation follows [48]:

Label: A switching protocol named multi-protocol label switching (MPLS) [49] is employed in the reduction of the packets processing time as it does not need each control packet to provide a fixed route instead the route is provided by the ingress node.
Wavelength ID: A single optical fibre has the capability to carry hundreds of individual wavelength channels and the control packet needs to be able to identify each channel, 16bits are allocated to allow the identification of these wavelengths.
Flag: The segment length field has a dynamic length so the flag identifies the beginning of the QOS indicator field and the end of the segment indicator field.

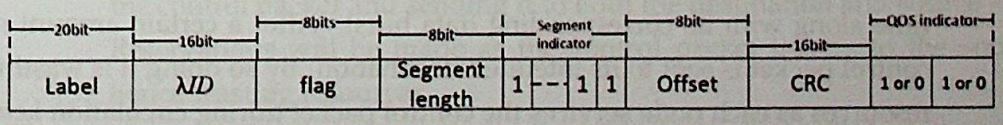


FIGURE 2.1: control packet structure

Segment Length (SL): This field contains the length of one segment in the control burst.

Segment Indicator(SI): The size of the segment indicator varies from one data burst to the other controlled by the burst assembly algorithm. The segment indicator identifies the maximum allowable bust segment size. To obtain the burst length core nodes multiply the SL by the number of ones in the SI.

Offset time: The offset time refers to the time gap between the control packet and the burst, and it's used to configure the switches before the burst reaches the core nodes.

CRC: For error detection a cyclic redundancy check is deployed which uses a hash function to detect any changes to data during transmission.

QOS Indicator: There are four qualities of service strategies that can be used in OBS [50]. These strategies are:

- No QOS requirements
- Packets are arranged in decreasing priority order.
- Priority packets is organized in ascending order.
- Higher priority packets are enveloped by lower priority packets

The QOS indicator field identifies which QOS strategy is deployed.

2.2 Security Threats in OBS

Optical burst switching can transfer data at tremendous speeds but like any other networking technique it suffers from some security vulnerabilities [7]. Sreenath et al [46] conducted a research and came up with a security threat which they called the burst circulation attack. This attack delays the delivery of data burst using a cluster of compromised nodes. One node becomes a master node and the rest as slaves. The control packet is redirected amongst the compromised nodes for a period of time along with its corresponding data burst. After a certain amount of time the control packet is sent to its intended destination. By so doing it is wasting network resources as each node services the control packet during circulation keeping them from servicing other control packets. Authors in [45], proposed a threat in which the threshold value is altered either at the ingress node or egress node and named it the burstification threat. The egress node checks the length of the data burst against the length of the burst stored in the segment indicator field in the control packet and if they do not match the burst is dropped from the network.

Since a control packet undergoes OEO conversion at every intermediate node by modifying the destination field of the control packet an attacker can tunnel the data burst to a malicious destination. This type of attack is called a burst hijacking attack [51] and it compromises the availability of data.

Every network is susceptible to Distributed Denial of Service attacks (DDoS) [52]. The most common DDoS attack is the flooding attack. Flooding attacks can be classified into two categories:

1. Spoofed header packets
   Spoofed header packets relates to when an external packet is introduced into the network [52]. The most common threat is the fake header injection attack. This attack arises when an attacker inserts a malicious burst into the network to redirect bursts to fake destinations [7].

2. Legal header packets used inappropriately
   This is when a control packet is redirected to a different node or when it is duplicated [53]. The most common attacks are;

   - Replay attack
     This attack occurs when a compromised node copies and stores the control packet. The stored control packet is later introduced back into the network and wastes resources as the control packet makes reservations for a burst that does not exist [54].
   - Burst header flooding attack
     A duplicate of the control packet is stored by a compromised node and injects a number of these duplicate control packets back into the network at a later time. This then prevents the node from servicing legitimate requests as nodes are being flooded with resource requests [51], this attack is termed burst header flooding attack.
   - Land attack
     A node can send a burst to multiple destinations on any wavelengths. Land attacks take advantage of this functionality by making a copy of the control packet and sending it to both the destination and source [46]. Reservations will be made as the control packet travels to the source hence wasting resources.

- Burst duplication attack
  This attack happens when a copy of a control packet is made by a compromised node and is sent to an attacker. The original control packet is sent to its intended destination [55].

2.3 Security Threat Detection Methods

Threat detection methods can be categorized into three types as shown by table 2.1 below.

TABLE 2.1: Identification methods

Detection type	Studies
statistical detection	[56], [51]
message authentication function	[57], [58], [59]
Machine learning	[60], [61], [62], [63]

All core nodes are monitored by trusted nodes in the network. These trusted nodes determine the presence of a malicious node through analyzing statistical data collected during transmission. Once a control packet is received by a core node, the node transmits certain statistical data to trusted nodes in a network. This data includes the destination and source addresses, burst id, and how many packets are contained within the burst. Upon receiving this data the trusted node inserts this data into a buffer table and compares it to data sent by adjacent nodes servicing the same control packet. If any node sends mismatching data then the node transmitting to it is deemed compromised. This detection method was proposed by the authors in [56] and [51].

Merlin *et al* [57] proposed a system in which they perform per hop authentication between nodes through a modified version of the HMAC algorithm. The MD5 algorithm is vulnerable to attacks, therefore, to mediate this they used a modified version of the MD5 algorithm to produce the message digest of the control packet. The message digest is then appended to the control packet and both are encrypted using the RC4-4S encryption scheme. Every node produces a message digest of the control packet and compares it to the appended message digest, if they vary then the transmitting node is deemed compromised.Another per hop authentication scheme proposed by Chen [58], in which MD5 is used to authenticate the control packet at each node. If authentication fails then the transmitting node is deemed compromised.

Control packet flooding attack is one of the most dominant attacks in OBS as such [60], [61] and [62] produced methods of identifying a flooding attack using different machine learning techniques. A system that monitors edge nodes based on automated classification of systems to determine a malicious edge node was developed by [62]. These systems used data mining techniques especially rule induction to determine compromised edge nodes.To detect flooding attacks, Kaur et al [61] developed a genetic algorithm to monitor edge nodes to determine compromised nodes. Zahid et al [60] proposed a model that could detect compromised edge nodes. This model utilized Deep Convolution Neural Network (DCNN) to automatically detect compromised edge nodes. Burst header flooding attacks drastically reduces the quality of service as resources are held up servicing non-existent requests. Authors in [63] developed a mechanism that utilized predictive machine learning to deal with burst header flooding attacks.

2.3.1 Evaluation metrics of threat detection methods in Optical Burst Switched networks

The identified methods only had one metric and that was the successful identification of a compromised node. Methods using message authentication functions have added metrics that were explored. However, all those methods primary objective was not to detect the existence of a threat in a network but to provide a preventative measure and detection arose as a byproduct. As such their metrics are covered in section 2.4 and will be explored at a later stage. Some methods of identification while prevalent in literature have no forms of validation specified by their respective authors [58], [61]. Two types of validation techniques were used to check the accuracy of the identified detection methods, these validation techniques being simulation and the use of a confusion matrix;

2.3.1.1 Simulation

A number of simulators were used to evaluate identified threat detection methods. A brief description of the simulators used by the identified studies follows along with the classification of the studies in accordance with their respective simulators given in table 2.2.

TABLE 2.2: simulators used for detection

Simulation software	Studies
Network simulator 2 (NS-2)	[46], [51]
xilinx ISE	[64]
NCTUns	[59], [63]

Network Simulator 2 (NS2)

Network simulator version 2 (NS-2) is a unix based discrete event driven simulator that is open source for research into networking. It allows the simulation of multicast protocols, TCP and routing over a wired and wireless network [65]. NS-2 does not support OBS simulation and needs an nOBS extension which allows the simulation of TCP over OBS in ns-2 [66].

NCTUns

This is both a network simulator and emulator supporting a variety of network protocols, applications and models for both wired and wireless networks. This simulator can be used to evaluate the performance of network application programs, network research education and network planning [65].

Xilinx ISE

Xilinx integrated synthesis environment (ISE) is used to compile and analyze hardware design language designs. This allows a researcher to determine how their design will react to different situations and view RTL diagrams[25].

2.3.1.2 Confusion Matrix

A confusion matrix shows how a model is confused when making predictions as it is a summary of prediction results. Predictions are categorized into different classes along with their count values making it possible to identify the types of errors being made by the classifier. Studies in [60] and [62] employed confusion matrix to validate their methods.

2.4 Security Threat Handling Methods in OBS

There are generally two methods to deal with security threats in OBS networks these methods are nodal isolation and cryptographic technique. These methods are outlined as follows:

The Nodal Isolation method

Nodal isolation is when a node is excluded from the network as no traffic is directed towards it. Once a security threat has been detected in a network and the compromised node identified, the compromised node is excluded from the network and no traffic is directed towards it [51], [61], [52], [62], [46], [63], [60].

Cryptographic Methods

The main mandate of cryptography is to secure communications by allowing the contents of the control packet to be viewed by both the sender and the intended recipient only. [67], [68]. This is achieved by encrypting the control packet using a private key and transmitting both the cipher-text and the private key to the recipient for decryption. Cryptographic techniques identified in the explored studies include Rivest Shamir Adleman (RSA), Hash-based Message Authentication Code (HMAC) and Quantum Cryptography and these are explained as follows:

Rivest–Shamir–Adleman (RSA)

RSA is a public cryptographic scheme whereby a transmitter encrypts data with the receivers public key and the receiver de-crypts their data with their secret key [69], [70], [71]. Multi layered security was employed to increase control packet security [72]. The transmitting node generates its temporary key pairs and sends its public key to its neighboring nodes. The transmitting node then generates its second key pairs and then encrypts its new public key with its old public and private keys. Upon receiving the new encrypted public key of the transmitter the receiving node initially de-crypts it with the receiver's old private key, and then with the transmitter's old public key.

Hash-based message authentication code (HMAC)

The Burst Control Packet is authenticated by using a modified HMAC algorithm which is termed the Enhanced HMAC (EHMAC) algorithm [64]. Instead of using the conventional MD or SHA hashing functions normally associated with the HMAC algorithm the modified MD5 algorithm is used. The message digest generated by the EHMAC is concatenated with the burst control packet and encrypted using the RC4-4S symmetric algorithm. The Burst Control Packet is then transmitted to the receiving node. The burst control packet is then de-crypted by the use of the RC4-4S decryption algorithm and the EHMAC authentication performed. The

generated message digest is compared to the message digest generated by the transmitting node if any irregularities are identified the control packet is dropped.

Quantum Cryptography

Quantum Cryptography is the application of quantum mechanics principles to produce a cipher-text. Quantum cryptography is based on Heisenberg's uncertainty theory, which states that certain pairs of physical properties cannot be determined at the same time [73]. One is computed, while the other is distributed and, as a result, difficult to compute. Quantum keys are transmitted on a channel separate from the data burst and sends along Q-bits which are photons of random polarization. As photons change when they are measured if an intruder attempts to intercept this data it can be detected.

Table 2.3 summarizes the main features of security threat handling methods in OBS networks.

TABLE 2.3: Features of threat handling methods

Nodal Isolation	**Cryptographic Techniques**
Per hop authentication is used to determine compromised nodes in the network	Compromised nodes are not identified in the network. Control packets are encrypted at all nodes in the network
Control packets are not directed towards compromised nodes in the network	Control packets are directed towards compromised nodes
Reduces network resource utilization as compromised nodes are not used	All resources in the network are utilized
Increases end-to-end delay as the control packet is rerouted in the network and ends up taking a longer path	Increases end-to-end delay as processing delays are increased at all intermediate nodes by the cryptographic algorithms

2.4.1 Evaluation Metrics Used to Evaluate Threat Handling Mechanisms in OBS Networks

All the security mechanisms deploying nodal isolation are only concerned with the successful exclusion of a compromised node from the network, they did not take

into account any other metric. Authors in [54] proposed a method to embed a security framework which utilizes Cryptography, both classical and quantum techniques into an OBS network architecture to enhance security. However, these authors did not validate their proposal in any way. Evaluation methods of identified studies are discussed below.

Secure Burst Control Packet Scheme for Optical Burst Switching Network [59]

Coulibaly *et. al* [59] designed a mechanism in which they used Rivest, Shamir, and Adleman (RSA) public key encryption algorithm which was assessed using the NCTUns network simulator. They looked at three scenarios which are: a network with no security measures and no security attack, a network under data burst redirection attack having no security measures and a network under data burst redirection attack with an RSA based security mechanism. In all these three cases they were comparing the throughput and burst loss ratio.

Enhanced HMAC Structure Based Burst Header Authentication Design for Optical Burst Switched Networks [57]

A mechanism in which an enhanced HMAC structure validated the authenticity and integrity of a data burst was developed by [57]. RC4-4S encryption was used to ensure the confidentiality of the control packet while a modified version of the MD5 algorithm was used to determine its authenticity. The mechanism was synthesized using the XILINX ISE such that it could be implemented in commercial off the shelf field-programmable gate array (FPGA) and hence be used in real-time OBS networks. Their main considerations were the avalanche effect, speed and to determine its effectiveness in real time applications.

2.5 Conclusion

In the consulted literature, control packet security mechanisms fall into a spectrum of two types: one that identifies a compromised node and isolate it from the network, and those that apply security protocols regardless of whether a threat exists or not. Both these approaches have major disadvantages. The isolation of a node from a network reduces network resource utilization while the application of security protocols throughout the network increases the end-to-end delay of both the

control packet and the data burst. The proposed work intends to formulate a mechanism that will ensure security, maximum utilization of resources in the network without major penalties of end-to-end delay in the network.

Chapter 3

Methodology

3.1 Introduction

This segment offers an in-depth look into the methodology to be followed to formulate an algorithm for OBS networks solely focused on the security of the control packet, maximizing resource utilization and reducing end-to-end delay. The secure packet mechanism is formulated by; analyzing and selecting a suitable control packet authentication technique for per hop authentication, analysis and selection of a security protocol method that ensures the confidentiality of the control packet throughout the networks and finally the formulation of the proposed secure control packet algorithm.

3.2 Control Packet Authentication

Message authentication functions are the most prominent way to ensure the integrity of data. There are two types of message authentication functions that identifies if the received data is identical to the data sent: namely Message Authentication Codes and Hash Functions. Message Authentication Codes create a cryptographic checksum by using a secret key. The checksum is then appended to the message [74]. This is only viable when two communicating parties have a common key. When one party wants to transmit a message to the other a checksum/MAC is produced as a function based on the key and the message [75] [74] [76]. Message Authentication Codes are used to safeguard the integrity and authenticity of data as the secret key is used to assure the receiver where the message originated as well as the fact that the message was not tampered with in any way.

Hash functions produce a hash code or hash which is a string of fixed length from a message of variable length. The hash is a laconic representation of the message being hashed [77]. Hash functions are used to guarantee data integrity meaning

that hash codes are generated without any external input and are only used to determine if a message was altered in any way during transmission [78]. This research focuses on the integrity of the data, hence use of a hash function to authenticate the integrity of the control packet through the core nodes. The next subsection explores Hash functions and focuses on their construction.

3.2.1 Hash functions

In cryptography a Hash function H is a function that produces a fixed length string from a string of random bit length, the output of which is termed a digest, hash value or hash code [79] [80] [81]. For a hash function to qualify as a cryptographic hash function it should be a one way function. That is, it should be mathematically impossible to obtain any value a in the domain H from any string b from a codomain H such that

$$H(a) = b \tag{3.1}$$

Hash functions do not have any key, but to ensure that they are secure they have three properties that they must possess [78]. These properties are as follows;-

1. Preimage resistance
   Hash functions must always be one way functions. That is, if you're given a hash value y it must be mathematically impossible to find the original message m.

2. Second preimage (weak collision) resistant
   No two messages should have the same hash value as such it should be computationally improbable for two messages $m_1 \neq m_2$ to produce the same hash values y_1=h(m_1) = h(m_2)= y_2.

3. Collision (Strong collision) resistant
   It should be mathematically impossible to obtain two colliding inputs. No two arbitrary messages should have the same hash value. That is for any hash function h, it should be impossible to find any two inputs r and j such that h(r) = h(j). Strong collision resistance applies to any two arbitrary messages while weak collision resistance applies to two particular messages.

The following section looks at how hash functions are constructed such that they posses the three properties required for hash functions to be considered secure.

Construction of Hash Functions

There are two methods of constructing a Hash function that is resistant to collision. One such method is the use of a compression function for short messages to build a hash function that is resistant to collision . The second method is to extend the collision resistant hash functions for short messages to build a collision resistant hash function for large messages [82] [80]. The most popular paradigm for archiving this is through the use of the Merkel Damgard paradigm.

The Merkel Damgard construction was invented by Merkel [83]and Damgard [80] in 1989 independently. Message Digest 4 (MD4) [84], Secure Hash Algorithm 0 (SHA-0) [85], Message Digest 5 (MD5) [86] , Secure Hash Algorithm 1 (SHA-1) [87], Secure Hash Algorithm 2 (SHA-2) [88] and Race Integrity Primitives Evaluation Message Digest 160 (RIPEMD-160) [89], all use the iterative nature of the Merkel Damgard method. The most important aspect of the Merkel Damgard method is the collision resistant compression function that generates a hash value of a fixed length from a fixed input length message.

A chaining variable and a message block are the two inputs of a compression function. The inputs of a compression function are an n-bit message block along with an m-bit chaining value such that $f : \{0,1\}^n * \{0,1\}^m -> \{0,1\}^n.h\{0,1\}^* -> \{0,1\}^m$ is the output of the iterative compression function algorithm obtained by processing a message of random size through the compression function [90]. Padding is always performed on the message that is to be processed such that the message becomes divisible by the original message block represented here by n bits. Bit-length n corresponds to the input bit length of the compression function f that is currently in use. A compression function processes a message of fixed length. If the message block n does not satisfy the length requirement then padding is performed by adding a single positive bit, and then a string of negative bits until the required length is reached [27]. To strengthen the Merkel-Damgård paradigm, the padding contains the ASCII representation of the original message. The message is then divided into blocks all consisting of n-bit length. After that, the hash function h can be deduced as follows;

$$\begin{aligned} Q_0 &= IV \\ Q_i &= f(Q_{i-1}, M_i), i = 1.....t \\ h(m) &= Q_t \end{aligned} \tag{3.2}$$

Where

- f is referred to as the compression function of the hash function h
- Q_i is just the intermediate chaining variable that lie between between the initial and final stage
- Q_0 is the first hash state value (IV).

The hash state value determines the computation of the hash value. When hashing begins there is a predefined value of the hash state values specified. The iterative nature of the Merkel Damgard construction using a compression function is depicted in Figure 3.1.

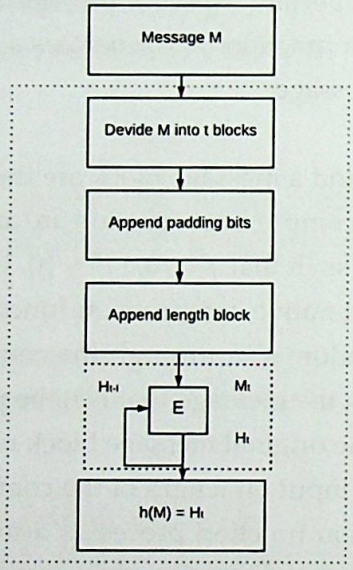


FIGURE 3.1: View of the Merkel Damgard construction reproduced from [91]

The hash state values are updated after every compression function action , until the whole message has been processed. The final hash state values become the hash code of the initial message. The Merkel Damgard construction is used to extend the collision resistant property of hash functions for short messages to build one for long messages. However, for the Merkel Damgard construct to be effective, a compression function for short messages must be developed.

MD-5, SHA-1, SHA-2 and RIPEMD-160 are among the commonly used hash functions and they are all derived from the Merkle-Damgård structure by utilizing a static compression function. The output of the MD-5 algorithm consists of 128-bits, while the output of both RIPEMD-160 and SHA-1 consists of 160 bits. SHA-2 consists of a triad of algorithms, SHA-256, SHA-384 and SHA-512, all with different outputs. SHA-256 has an output of 256-bits, where as SHA-384 has an output of 384-bits and finally SHA-512 has 512-bits. Each and every one of these hash functions originate from the MD-4 algorithm. The MD-4 algorithm is the basis in which most hash functions are derived and is discussed below to comprehend the origin of the other hash functions as it is the simplest of all the constructions. The exploration of the MD4 algorithm is then followed by an in-depth look of all the algorithms making up the MD4 family.

3.2.1.1 MD4

MD-4 consists of three bitwise functions operating on three 32-bit variables.

$$f(x,y,z) = (x \wedge y) \vee ((\neg x) \wedge z) \tag{3.3}$$

$$g(x,y,z) = (x \wedge y) \vee (x \wedge z) \vee (y \wedge z) \tag{3.4}$$

$$h(x,y,z) = x \oplus y \oplus z \tag{3.5}$$

Throughout the algorithm a current hash state is maintained, corresponding to the internal state variable used to store the hash value as processing takes place. This hash state is denoted by

$$Q = (Q_1, Q_2, Q_3, Q_4)$$

with each value consisting of 32-bit. The final length of the MD-4 is 128 bits [90]. The length of the message fed into the compression function is 640 bits, it is made up of two parts the chaining variable of 128 bits and the message block being hashed of 512 bit. There are three other constants (q_i, w_i, e_i) being used in the MD-4 algorithm, which varies depending on the round. These constants are

$$q_j = \begin{cases} 0 & 0 \leq j \leq 15, \\ 0x5A827999 & 16 \leq j \leq 31, \\ 0x6ED9EBA1 & 32 \leq j \leq 47. \end{cases} \tag{3.6}$$

the arrays below provide values of e_i and w_i,

$$
\begin{aligned}
w_{0...15} &= [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15],\\
w_{16...31} &= [0,4,8,12,1,5,9,13,2,6,10,14,3,7,11,15],\\
w_{32...47} &= [0,8,4,12,2,10,6,14,1,9,5,13,3,11,7,15],\\
e_{0...15} &= [3,7,11,19,3,7,11,19,3,7,11,19,3,7,11,19],\\
e_{16...31} &= [3,5,9,13,3,5,9,13,3,5,9,13,3,5,9,13],\\
e_{32...47} &= [3,9,11,15,3,9,11,15,3,9,11,15,3,9,11,15].
\end{aligned}
$$

The procedure outlined in Algorithm 1 is then executed for every message block entered into the compression function after padding has been performed. Bit wise rotation to the left is denoted by $\ll$. On each iteration 512 bits of the message block are loaded into X_j for $0 \leq j < 16$. Since the hash state values are subjected to change as MD4 is processed temporary hash state variables are used to store the hash variables as they are manipulated and updated. These temporary hash variables are denoted by R, J, K and L.

Algorithm 1 MD-4 compression function reprinted from [90]

(R,J,K,L)$\leftarrow IV_{r-1}= (Q_1, Q_2, Q_3, Q_4)$
/* ROUND 1 */
for j = 0 to 15 **do**
    t $\leftarrow R + f(J,K,L) + X_{ej} + q_j$
    (R,J,K,L) $\leftarrow$ (L, t $\ll e_j$,J, K).
/* ROUND 2 */
for j = 16 to 31 **do**
    t $\leftarrow A + g(J,K,L) + X_{wj} + q_j$
    (R,J,K,L) $\leftarrow$ (L, t $\ll e_j$,J, K).
/* ROUND 3 */
for j = 32 to 47 **do**
    t $\leftarrow R + h(J,K,L) + X_{wj} + q_j$
    (R,J,K,L) $\leftarrow$ (L, t $\ll e_j$,J,K).
$(Q_1, Q_2, Q_3, Q_4) \leftarrow IV_r = (Q_1 + R, Q_2 + J, Q_3 + K, Q_4 + L)$

The final hash state values are then linked together and taken as the hash value of the MD-4 algorithm. Initially the hash state values are initialized with the following

values;

$$Q_1 = 0x67452301$$
$$Q_2 = 0xEFCDAB89$$
$$Q_3 = 0x98BADCFE$$
$$Q_4 = 0x10325476$$

3.2.1.2 MD5

Just like MD-4, MD-5 produces a hash value of 128-bits from a message of 512-bit blocks [79]. The difference between the two is that in addition to the f, g and h functions of MD-4, MD-5 has an additional function i defined as

$$i(x, y, z) = y \oplus (x \vee (\neg z)) \tag{3.7}$$

and the function of g has been changed to

$$g(x, y, z) = (x \wedge z) \vee (y \wedge (\neg z)) \tag{3.8}$$

to make g's structure less symmetric. There are 64 predefined key values $K_0, ..., K_{63}$ one for each operation. Bitwise rotation is performed on each operations. The bits need to be rotated left by a predefined values stated below;

$$s[0..15] := 7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22$$
$$s[16..31] := 5, 9, 14, 20, 5, 9, 14, 20, 5, 9, 14, 20, 5, 9, 1, 20$$
$$s[32..47] := 4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23$$
$$s[48..63] := 6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21$$

The hash state variables are four 32-bit values $Q = (Q_1, Q_2, Q_3, Q_4)$ giving an output of 128-bits from input blocks of 512-bits. The message is loaded as 16 words/512-bits at a time into X_j for $0 \leq j < 16$ which are at that point extended to 64 words. After the entire message is processed the final chaining variables are combined to produce a hash value that represents the original message [79] as shown in algorithm 2.

Algorithm 2 MD-5 compression function adapted from [90]

(R,J,K,L)$\leftarrow IV_{r-1}= (Q_1, Q_2, Q_3, Q_4)$
/* ROUND 1 */
for j = 0 to 15 **do**
    t $\leftarrow J + f(J,K,L) + X_{zj} + k_j$
    (R,J,K,L) $\leftarrow$ (L, t $\ll s_j$,J, K).
/* ROUND 2 */
for j = 16 to 31 **do**
    t $\leftarrow J + g(J,K,L) + X_{zj} + k_j$
    (R,J,K,L) $\leftarrow$ (L, t $\ll s_j$,J,K).
/* ROUND 3 */
for j = 32 to 47 **do**
    t $\leftarrow J + h(J,K,L) + X_{zj} + k_j$
    (R,J,K,L) $\leftarrow$ (L, t $\ll s_j$,J,K).
/* ROUND 4 */
for j = 48 to 63 **do**
    t $\leftarrow J + i(J,K,L) + X_{zj} + k_j$
    (R,J,K,L) $\leftarrow$ (L, t $\ll s_j$,J,K).
$(Q_1, Q_2, Q_3, Q_4) \leftarrow IV_r = (H_1 + R, H_2 + J, H_3 + K, H_4 + L)$

The initial chaining values of 128-bit values to be processed with the first message blocks are given as follows;

$$Q_1 = 0x67452301$$
$$Q_2 = 0xefcdab89$$
$$Q_3 = 0x98badcfe$$
$$Q_4 = 0x10325476$$

3.2.1.3 SHA-1

SHA-1 has five internal states rather than four 32-bit values in MD-4
$Q = (Q_1, Q_2, Q_3, Q_4, Q_5)$ resulting in a hash value of a 160 bits from a message block of 512 bits [92]. A message is broken down into blocks of 448 bits which are then padded with 64 bits to make their overall length 512 bits. SHA-1 is extremely similar to MD4 as they use identical bit-wise functions. These functions f, g and h. SHA-1

has four round constants being:

$$k_1 = 0x5A827999$$
$$k_2 = 0x6ED9EBA1$$
$$k_3 = 0x8F1BBCDC$$
$$k_4 = 0xCA62C1D6$$

after the message has been padded. The message is then fed 16 words/512 bits one after the other into X_j for $0 \leq j < 16$. SHA-1 uses a more extensive version of X_j spanning from 0 to 79 as SHA-1 has 80 rounds. Steps in algorithm 2 are then applied to each of the 16 words. After all the data is executed the output is the concatenation of the final hash state variables those being Q_1, Q_2, Q_3, Q_4, Q_5 as depicted in algorithm 3. SHA-1 has an additional hash state as a result one more temporary hash state is introduced to accommodate the extra hash state. As such M is added to the already existing R, J, K, L temporary hash states.

Algorithm 3 SHA-1 compression function obtained from [90]

(R,J,K,L,M)$\leftarrow IV_{r-1}= (Q_1, Q_2, Q_3, Q_4, Q_5)$
/* Expansion */
for j = 16 to 79 **do**
    t $X_j \leftarrow ((X_{j-3} \oplus X_{j-8} \oplus X_{j-14} \oplus X_{j-16}) \lll 1)$
/* ROUND 1 */
for j = 0 to 19 **do**
    t= (R$\lll$5)+f(J,K,L)+M+X_j+k_1
    (R,J,K,L,M) $\leftarrow$ (t,R,J $\lll$ 30,K,L).
/* ROUND 2 */
for j = 20 to 39 **do**
    t= (R$\lll$5)+h(J,K,L)+M+X_j+k_2
    (R,J,K,L,M) $\leftarrow$ (t,R,J $\lll$ 30,K,L)
/* ROUND 3 */
for j = 40 to 59 **do**
    t= (A$\lll$5)+g(J,K,L)+M+X_j+k_3
    (R,J,K,L,M) $\leftarrow$ (t,A,B $\lll$ 30,C,D)
/* ROUND 4 */
for j = 60 to 79 **do**
    t= (A$\lll$5)+h(J,K,L)+M+X_j+k_4
    (R,J,K,L,M) $\leftarrow$ (t,R,J $\lll$ 30,K,L)
$(Q_1, Q_2, Q_3, Q_4, Q_5) \leftarrow IV_r = (Q_1 + R, Q_2 + J, Q_3 + K, Q_4 + L, Q_5 + M)$

The initial hash state values of 160 bit values to be processed with the first message blocks are initially given as:

$$Q_1 = 0x67452301$$
$$Q_2 = 0xEFCDAB89$$
$$Q_3 = 0x98BADCFE$$
$$Q_4 = 0x10325476$$
$$Q_5 = 0xC3D2E1F0$$

3.2.1.4 SHA-256

SHA-256 produces a hash value of 256 bit hash value. SHA-256 varies varies from MD-4 and SHA-1 as it employs different f and g functions. These functions being:

$$f(x,y,z) = (x \wedge y) \oplus ((\neg x) \wedge z) \tag{3.9}$$

$$g(x,y,z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) \tag{3.10}$$

The following functions are also performed on single 32-bit words

$$\gamma_0(z) = (z \ggg 2) \oplus (z \ggg 13) \ominus (z \ggg 22) \tag{3.11}$$

$$\gamma_1(z) = (z \ggg 6) \oplus (z \ggg 11) \ominus (z \ggg 25) \tag{3.12}$$

$$\delta_0(z) = (z \ggg 7) \ominus (z \ggg 18) \oplus (z \gg 3) \tag{3.13}$$

$$\delta_1(z) = (z \ggg 17) \ominus (z \ggg 19) \oplus (z \gg 10) \tag{3.14}$$

where $\ggg$ is the right rotate and right shifting is denoted by $\gg$. SHA-256 has 64 words $K_0, ..., K_{63}$ which are obtained from the first 64 prime numbers as the first 32 bits of their cube roots' fractional component. The hash state variables are eight 32-bit values $Q = ((Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7, Q_8))$ giving an output of 256-bits from input blocks of 512-bits. The message is loaded as 16 words/512-bits at a time into X_j for $0 \leq j < 16$ which is then extended to a total of 64 words. After the entire message is processed the final chaining variables are added together to get the initial message's hash value. SHA-256 uses eight temporary hash state values denoted by R, J, K, L, M, N, O, P.

Algorithm 4 SHA-256 compression function reprinted from [90]

(R,J,K,L,M,N,O,P)$\leftarrow IV_{r-1} = ((Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7, Q_8))$
/* Expansion */
for j = 17 to 64 **do**
  t $X_j \leftarrow \delta_1 (X_{j-2}) + X_{j-7} + \delta_0(X_{j-15}) + X_{j-16}$
/* ROUND */
for j = 1 to 64 **do**
  $t_1 = \leftarrow$ H $+\gamma_1(M) + f(M,N,O) + K_i + X_i$
  $t_2 = \gamma_0(R) + g(R,J,K)$
  (R,J,K,L,M,N,O,P)$\leftarrow (t_1 + t_2, R, J, K, L + t_1, M, N, O)$
$((Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7, Q_8)) \leftarrow IV_r = (Q_1 + R, Q_2 + J, Q_3 + K, Q_4 + L, Q_5 + M, Q_6 + N, Q_7 + O, Q_8 + P)$

The initial hash state values of 256-bit values to be processed with the first message blocks are given as follows:

Q_1 = 0x6A09E667, Q_2 = 0xBB67AE85,
Q_3 = 0x3C6EF372, Q_4 = 0xA54FF53A,
Q_5 = 0x510E527F, Q_6 = 0x9B05688C,
Q_7 = 0x1F83D9AB, Q_8 = 0x5BE0CD19.

3.2.1.5 RIPEMD-160

The hash function RIPEMD-160 compresses any arbitrary message less than 2^{64} length into a hash value with length of 160-bit [89]. Padding is performed on the message, the message is then processed as blocks consisting of 512 bits following the iterative design of the Merkle-Damgård construct. RIPEMD-160 produces a message digest of 160 bits. The compression function has two parallel operations: left branch and right branch. Each branch contains 5 rounds, and each round contains 16 steps [81] [93]. The Boolean functions in each round are as follows:

$$q_1(a,b,c) = a \oplus b \oplus c \tag{3.15}$$

$$q_2(a,b,c) = (a \wedge b) \vee ((\neg a) \wedge c) \tag{3.16}$$

$$q_3(a,b,c) = (a \vee \neg b) \oplus c \tag{3.17}$$

$$q_4(a,b,c) = (a \wedge c) \vee (b \wedge \neg c) \tag{3.18}$$

$$q_5(a,b,c) = a \oplus (b \vee \neg c) \tag{3.19}$$

The input message is divided into 16 words m_k all consisting of 32-bit each [94] [89]. The expanded message words used to update the right branch are denoted by $m_{k(r)}$, while the message block used to amend the left branch is denoted by $m_{k(l)}$. The initial chaining variables of RIPEMD-160 compression function are given by $H = (Q_1, Q_2, Q_3, Q_4, Q_5)$. The compression function has 80 rounds that run twice in two parallel branches hence chaining variables for both branches are defined as follows:

$$R_0 = L_0 = H_2$$
$$R_{-1} = L_{-1} = H_3$$
$$R_{-4} = L_{-4} \ggg 10 = H_1$$
$$R_{-3} = L_{-3} \ggg 10 = H_5$$
$$R_{-2} = L_{-2} \ggg 10 = H_4$$

R denoting the right branch while L denotes the left branch. The state update transformation starts from a 160-bit input chaining variable $H = (Q_1, Q_2, Q_3, Q_4, Q_5)$ and updates them in 80 steps (5 rounds, each round contains 16 steps). In round $j(1 \leq j \leq 5)$, R_i and $L_i(1 \leq i \leq 80)$ are updated as follows:

$$R_i \leftarrow ((R_{i-4} \lll 10) + (R_{i-5} \lll 10) + q_j(R_{i-1}, R_{i-2}, (R_{i-3} \lll 10)) + m_{k(r)} + k_{j(r)}) \lll s_{i(r)}$$

$$L_i \leftarrow ((L_{i-4} \lll 10) + (L_{i-5} \lll 10) + q_{6-j}(L_{i-1}, L_{i-2}, (L_{i-3} \lll 10)) + m_{k(l)} + k_{j(l)}) \lll s_{i(l)}$$

s_i denotes the shift position. Once the entire message is processed the final chaining values becomes the message digest of the original message. The final chaining variables are updated as follows:

$$\begin{aligned}
H_1 &= H_2 + R_{79} + (L_{78} \lll 10) \\
H_2 &= H_3 + (R_{78} \lll 10) + (L_{77} \lll 10) \\
H_3 &= H_4 + (R_{77} \lll 10) + (L_{76} \lll 10) \\
H_4 &= H_5 + (R_{76} \lll 10) + (L_{80}) \\
H_5 &= H_1 + (R_{80}) + (L_{79})
\end{aligned}$$

3.2.2 Speed and Security of Hash Functions

To compare the speed of the various hashing functions single step computation is used. Single step computation refers to the number of operations carried out by a hashing algorithm on a single message block [81]. The less amount of operations an algorithm has to compute during processing the faster it is going to be. However, security of hashing algorithms lies in their ability to obscure the plaintext and this is achieved by the operations it carries out. The more operations carried out the higher the security of the hashing algorithm.

SHA-1

Step-by-step procedure of SHA-1 comprises of:

- four additions
- two shifts
- one Boolean function

As shown in algorithm 3 and appendix A.1. Boolean functions in SHA-1 do not perform the same amount of unit operations. f performs 4 unit operations, g performs 5 unit operations and finally h performs 2 unit operations. The number of unit operations performed on one message block amounts to:

step operations for Boolean function f = 4+2+4=10
step operations for Boolean function g = 4+2+5=11
step operations for Boolean function h = 4+2+2=8

$$Unit_operations = steps * step_operations = (20 * 10) + 2(20 * 8) + (20 * 11)$$
$$= 740(unit_operations)$$

SHA-256
The step operation of SHA-256 comprises of:

- seven addition operation
- two summation operations each consisting of five unit operations
- two Boolean function one consisting of five unit operations and the other of four

$$Step_operations = 7 + 2 * 5 + 5 + 4 = 26$$
$$Unit_operations = steps * step_operations = 64 * 26 = 1664(unit_operations)$$

The step operations of SHA256 are further illustrated in Appendix A.3 and Algorithm 4

MD5
As shown in Appendix A.2 and Algorithm 2 the step operation of MD5 consists of:

- 4 additions
- 1 shift
- 1 Boolean function

The Boolean functions have varying amounts of unit operations performed in each with f and g with 4 unit operations each,i with 3 unit operations and finally h with 2 unit operations. The number of unit operations amount to:

$$Unit_operations = steps * step_operations = (16 * 6) + (16 * 7) + 2(16 * 8)$$
$$= 464(unit_operations)$$

RIPEMD-160
The step operation of RIPEMD-160 consists of:

- 4 additions
- 4 circular shift
- 1 Boolean function

as depicted in Appendix A.4, q_1 consists of 2 unit operations, q_2 consists of 4 unit operations, q_3 consists of 3 unit operations, q_4 consists of 4 unit operations and q_5 consists of 3 unit operations hence the number of unit operations for RIPEMD-160 are given by

$$Unit_operations = steps * step_operations = 2((16 * 10) + 2(16 * 12) + 2(16 * 11)) \\ = 1792(unit_operations)$$

The more unit operations carried out by an algorithm the more secure it is. However, the speed reduces as the number of unit operations increases. This is because more time is used to carry out the operations. RIPEMD-160 is the most secure hash function however it is also the slowest due to the amount of operations that have to be carried out.The goal is to create an algorithm that will be secure and reduce end-to-end delay as such speed and security are the matrices to be considered when selecting the hashing algorithm. SHA-1 and MD5 are highly susceptible to collision attacks, RIPEMD-160 is slow. SHA-256 performs an average number of unit operations suggesting that it has the right blend of security and speed as such it is chosen to authenticate the control packet between nodes.

3.3 Security Protocol Method Selection

One method of securing information is through the use of cryptography which is the use of mathematical concepts to transform information into a format that is difficult to decipher. There are two types of cryptographic techniques: Symmetric cryptography which is also termed secret key algorithms involves using an identical key to perform both encryption and decryption. Asymmetric cryptography, also known as public key algorithms, employs two distinct keys: a public and a hidden key. Before transmitting the message the sender has to encrypt the message with the recipient's public key.The receiver decrypts the cipher text using his or her secret key.

From their inception asymmetric keys were never intended for data encryption but rather were developed as a means of a key transfer mechanism for symmetric cryptography schemes [95]. Public key cryptography requires the use of key management infrastructure so as to verify the keys. If this is not deployed, then attackers can

insert their own public keys and as such introduce man in the middle attacks, Public Key Infrastructure (PKI) is essential in symmetric key cryptography as it provides a platform for which keys can be safely transmitted in a network. Without a PKI, symmetric cryptography is no less secure than asymmetric cryptography. When implementing public key cryptography, the parties involved are to have the same initial parameters which are the numbers fed into the algorithm, this then leaves the algorithm vulnerable to evil values attacks which can substitute these numbers to disable encryption [96]. Symmetric encryption has a much simpler algorithm and hence its computationally faster than asymmetric cryptography [96]. Symmetric cryptography is faster [97] [98] than asymmetric cryptography, since the main consideration of this research is speed and security, one of the symmetric algorithm will be proposed for our solution.

3.3.1 Symmetric Algorithms

The Data Encryption Standard (DES) [99], Triple Data Encryption Standard (3DES) [100], The Advanced Encryption Standard (AES) [101], and Blowfish [102] are some of the symmetric algorithms. These algorithms are explored in the following sections.

3.3.1.1 Data Encryption Standard (DES)

DES encrypts blocks of 64-bit length and has a key of 56-bit. It utilizes an identical key for both encryption and decryption with each block undergoing 16 rounds of encryption. Each round performs the same operations on the block. The main 56-bit key is used to formulate 16 sub keys which are used during encryption rounds. Figure 3.2 depicts the block diagram of DES.

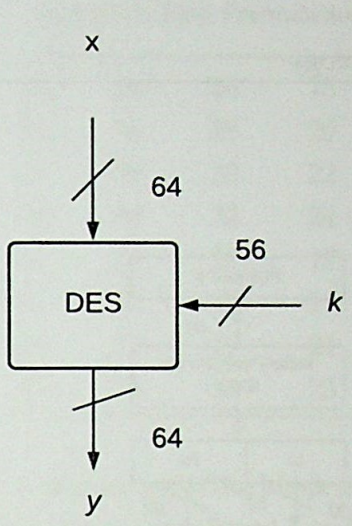


FIGURE 3.2: Input and output parameters of DES obtained from [27]

An initial permutation, 16 encryption rounds, and a final permutation make up the key structure of DES. DES uses a Feistel network which is iterative in nature and can employ a round function. DES also uses a non linear function *f*. The round function *f*'s input is a 48-bit round key derived from the main 56 bit key as well as two 32-bit cipher words (denoted by R and L) derived from the plaintext after undergoing initial permutation. The DES Feistel structure is shown in Figure 3.3.

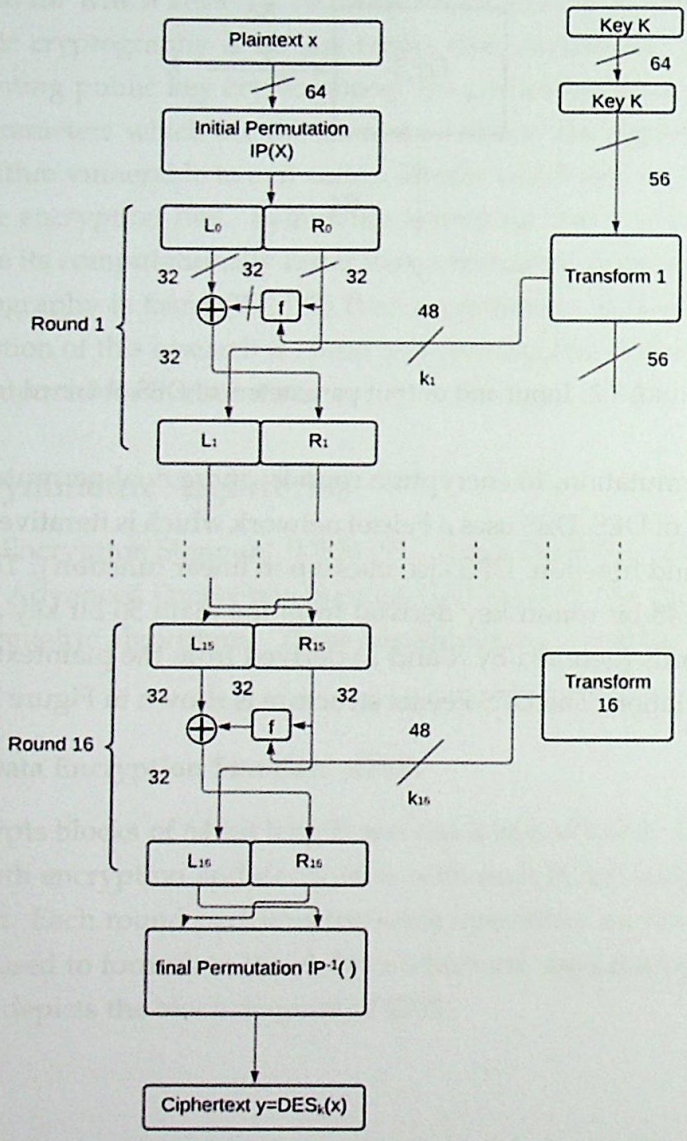


FIGURE 3.3: feistel structure of DES reprinted from [103]

The initial permutation is a bitwise permutation used to arrange the plaintext in such a way that it can easily be fetched by the 8-bit data busses [27]. Initial permutation arranges the plaintext in accordance with table 3.1. This shifting of data does not increase the security of the algorithm in any way as there is no key being used at this point.

TABLE 3.1: First Permutation

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Once the first permutation is carried out, The block is divided into two. A half on the right and a half on the left denoted by R and L in Figure 3.3.

The f function consists of four layers:

- Expansion layer:- The 32-bit cipher-word is extended to 48-bits by dividing it into eight blocks each consisting of 4-bits, each block is then expanded to 6-bits. The expansion table is provided in appendix B.9.
- Key addition:- After expansion takes place the 48-bit block is XORed with the 48 bit round key.
- Substitution layer:- The cipher-word is split into eight blocks each consisting of 6-bits. Blocks are fed into different substitution blocks that map the 6-bit blocks to 4-bit block to produce a 32-bit cipher-word. First and last bits denote the row and the four middle numbers denote the column.Once obtained the number is converted to binary and taken as the output of the substitution box. The substitution boxes are given in appendix B.
- Permutation layer:- The 32-bit output block of the substitution layer is permuted in accordance with table B.10 to introduce diffusion.

The XOR cipher is used to add the The right-hand side f(R_0, K_1) and the left-hand side L_0. This results in R_1 the first round output. The above process is repeated 15 times with varying round keys. The original R_0 block becomes the new $L1$ block in the second round with the results obtained from f(R_0, k_1)$\oplus$ L_0 becoming R_1. In more general terms the subsequent blocks are obtained through:

$$\begin{aligned} L_n &= R_n - 1 \\ R_n &= L_n - 1 + f(R_n - 1, k_n) \end{aligned} \tag{3.20}$$

Where:

- n is the round number
- L is the left block
- R is the right block
- k is the round key
- f is the compression function F

To obtain the ciphertext the original permutation performed on the data has to be reversed. This is done in accordance with the permutation table as seen in table 3.2.

TABLE 3.2: Final Permutation

40	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

The results of which are the ciphertext of the original data block. To decrypt DES the algorithm is run in the same exact way with one major difference, that is the round keys are used in reverse

3.3.1.2 Triple Data Encryption Standard (3DES)

Due to the advancement of computing power, a 56-bit key is now considered too short for modern applications. To avoid redesigning a completely new block cipher a cipher was proposed in which the key length of DES was increased. Triple DES operates by iterating DES three times using different keys, subsequently increasing the key size to 168-bits as shown in Figure 3.4. To encrypt the plaintext, the first key is used. The second key is used to decrypt the ciphertext while the third key is used to encrypt the second key's decrypted text. In all these three processes the DES structure outlined above is followed.

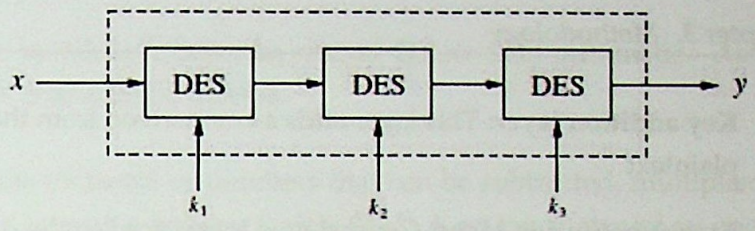


FIGURE 3.4: Triple DES structure obtained from [27]

$$3DES = DES_{K1} * DES_{K2} * DES_{K3}$$

Where K1, K2 and K3 donate the main keys of each DES application.

3.3.1.3 The Advanced Encryption Standard (AES)

Advanced Encryption Standard (AES) encrypts data of 128-bit block size. It's a non-Feistel cipher with key lengths ranging from 10, 12, or 14 rounds depending on the key length used [96]. For a length of 128 bits for the key AES has 10 rounds, with a 192-bits key length 12 round are required and 256-bit key length undergoes 14 rounds [96], [27] and [97]. Figure 3.5 displays a block diagram of the AES algorithm.

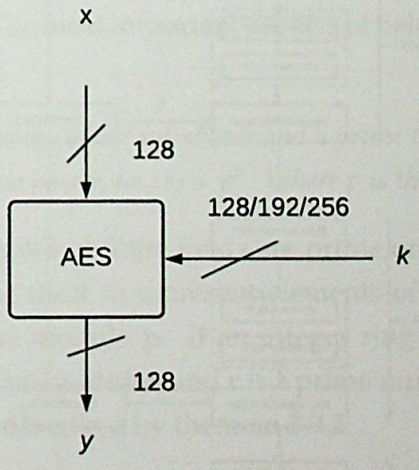


FIGURE 3.5: Input output parameters of AES reprinted from [27]

The encryption structure of AES has three layers with each layer manipulating all 128 bits of the plaintext. All rounds undergo all three layers except the first round. An explanation of the different layers follow with a more detailed illustration of the encryption process shown in Figure 3.6 [104]:

- **Key addition layer:** This layer adds a key derived from the original key to the plaintext
- **Byte substitution layer:** Confusion is introduced to the data through the use lookup tables. Each bit in the block is transformed in such a way that the changes propagate across all the block bits.
- **Diffusion layer:** This layer is divided further into two sub layers being the ShiftRow layer and MixColumn layers that are aimed at providing diffusion across all bits in the block. The ShiftRow layer consists of four rows, each with four bytes. The first row is not affected in any way while bits in the second row are right shifted three times, bits in the third row are right shifted twice, the fourth row is right shifted once [105].

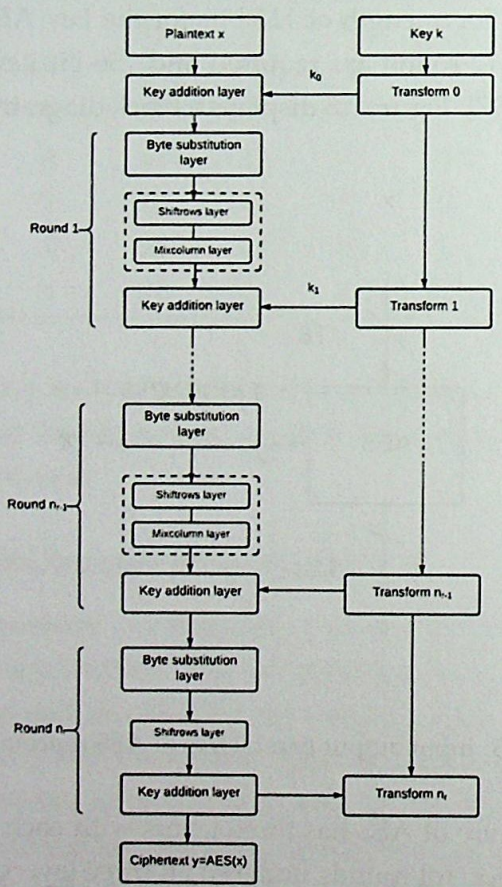


FIGURE 3.6: encryption structure of AES reprinted from [104]

Galois Fields

The *s*-boxes and MixColumn layer uses Galois field arithmetic. As such, a brief overview of Galois fields follows.

A set of finite elements or numbers that can be subtracted, multiplied, added and inverted are termed a finite or Galois field. A set consisting of an operation which has a corresponding inverse operation is referred to as a group, such that every element n has an inverse -n in such a way that $n + (-n) = 0 \cong 0$ mod m. A set containing additive and multiplicative groups is required for a structure to possess all four basic arithmetic operations. This structure is called a field. A field has three basic properties:

- .With the group operation "+", all elements inclusive in f, as well as the natural element *0*, form an additive group.
- A multiplicative group is formed by elements of f as well as the natural element 1 excluding *0*, this multiplicative group posses the group operation "×".
- The distributive law is always in effect when group operations are mixed, i.e., for all x,y,z ∈ F: x(y+z) = (xy)+(xz).

Cardinality or Order is the number of elements in a field [27]. Cryptography always deals with finite fields. The most important aspects of fields are provided in theorem 3.3.1 reprinted from [27].

Theorem 3.3.1. *For a positive whole number n and a prime whole number p, a field exists only if its order m is a prime power, i.e., m = p^n. Where p is the finite field's characteristic.*

The most basic examples of finite fields are prime order fields, fields with n=1. Integers 0,1,...,p-1 can be used to represent elements of the field GF(p). The two operators of the field are modulo p. If an integer ring $\mathbb{Z}_r$ contains integers with modular multiplication and addition and r is a prime number then $\mathbb{Z}_r$ is both a ring and a finite field as best described by theorem 3.3.2

Theorem 3.3.2. *The integer ring $\mathbb{Z}_p$ of a prime p is represented as GF(p) and is termed a prime field, or a Galois field consisting of elements of prime number. All elements of GF(p) other than zero have an inverse. All mathematical computations in GF(p) is done modulo p.*

To perform arithmetic operations on prime fields, rules of integer rings must be followed. These rules being;-

- Multiplication and addition are computed modulo p.

- The negation of every number an is calculated as follows: a+(-a)≅0 mod p.
- The reciprocal of any number not equal to zero is given by $a * a^{-1} = 1$.

AES uses the field GF(2^8) as its finite field contains 256 elements. Each byte of the internal data path is treated by AES as an element for the MixColumn and s-boxes. In the absence of a prime ordered finite field, modulo 2^8 addition and multiplication cannot be performed as 2^8 is not prime. Extension fields are fields with n > 1 and as such possess different notation to standard fields, and arithmetic performed on them is that of polynomials.

Polynomials with coefficients in GF(2) are used to represent GF(2^n) elements in extension fields. Every element has n coefficients meaning the polynomial has a degree of n-1. AES uses the field GF(2^8) which is represented in polynomial form as;

$$\begin{aligned} D(x) &= d_7x^7 + d_6x^6 + d_5x^5 + d_4x^4 + d_3x^3 + d_2x^2 + d_1x + d_0 \\ d_i &\in GF(2) = \{0,1\} \end{aligned} \tag{3.21}$$

Addition in extension fields is achieved by performing modulo 2 addition on coefficients with equal powers of x, while subtraction is performed by modulo 2 subtraction of coefficients with equal powers of x. MixColumn operations utilizing field multiplication in GF(2^8). As such two elements of a finite field GF(2^n) are multiplied following the standard multiplication rule for polynomials;

$$\begin{aligned} C(x) \cdot D(x) &= (c_{n-1}x^{n-1}.....c_0) \cdot (d_{n-1}x^{n-1}.....d_0) \\ E'(x) &= e'_{2n-2}x^{2n-2}.....e'_0 \end{aligned} \tag{3.22}$$

where;

$$\begin{aligned} e'_0 &= c_0d_0 mod 2 \\ e'_1 &= c_0d_1 + c_1d_0 mod 2 \\ e'_{2n-2} &= c_{n-1}d_{n-1} mod 2 \end{aligned} \tag{3.23}$$

Inversion of finite fields is a very important concept in AES as s-boxes utilize this property of finite fields during Byte substitution transformation [106]. For any irreducible reduction polynomial P(x) and its accompanying finite field GF(2^n), the inverse A^{-1} of a non-zero element A ∈ GF(2^8) is defined as;

$$A^{-1}(x) \cdot A(x) = 1 mod P(x) \tag{3.24}$$

Inversion lookup tables of fields with 2^{16} elements or less are often used. As previously stated AES uses GF(2^8) in which the inversion lookup table for all elements used are provided in appendix C.1. All inverses in GF(28) modulo P(x) = $x^8 + x^4 + x^3 + x + 1$ in hexadecimal form are listed in the table. For zero elements, there is no inverse. However, AES makes an exception and assumes that that the inverse of entry zero is mapped to itself.

Structure of AES

AES is a byte-oriented encryption algorithm, the 128-bit input block size is equivalent to 16 bytes [27]. The 16 bytes are fed into the s-boxes, the output of which are byte-permutated in the ShiftRow layer. The results obtained after the ShiftRow layer are jumbled up as a result of the MixColumn transformation [107], [105] and [27]. The bit-key is then added to the output of the MixColumn transformation layer [107], [105] and [27]. The Byte Substitution layer and Diffusion layer which constitutes the structure of AES are explained below.

Byte Substitution Layer

This layer has 16 identical *s*-boxes consisting of 8-bits. These S-boxes substitutes each state-byte by with another byte. The *s*-boxes can be reversed because to the fact that *s*-box substitution uses bijective mapping, meaning that every 256 possible entry element is mapped to one corresponding output element [27] and [90]. *s*-boxes are provided in 256 by 8 bit lookup tables with a set of fixed entries. The look up table is given in table 3.3.

TABLE 3.3: AES S-Box: Hexadecimal representation of substitution values obtained from [27]

		Y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
	1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
	2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
	3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
	4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
	5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
	6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
	7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
X	8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
	9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
	A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
	B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
	C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
	D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
	E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
	F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

S-boxes consists of two mathematical transformations; Galois field inversion and and affine mapping.

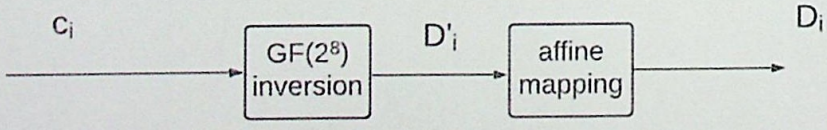


FIGURE 3.7: S-box mathematical operations

Galois field inversion is performed on each input element C_i. The inversion is given by $D_i' = C_i^{-1}$ and all the elements in C_i and D_i' are in the GF(2^8) field along with a fixed polynomial P(x)= $x^8 + x^4 + x^3 + x + 1$ [27] and [106]. The inversion table is provided in appendix C.1. The results obtained after inversion is performed are

multiplied by a constant matrix which is then added to an 8-bit vector, the operations of which are described as follows;

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ d_6 \\ d_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} d'_0 \\ d'_1 \\ d'_2 \\ d'_3 \\ d'_4 \\ d'_5 \\ d'_6 \\ d'_7 \end{bmatrix} mod2$$

$D'_i(x) = C_i^{-1}(x)$ is represented by a vector by $D' = (d'_7, d'_6, d'_5, d'_4, d'_3, d'_2, d'_1, d'_0)$. The above equation 3.3.1.3 is termed as affine mapping. Non-linearity is achieved in the S-boxes by the performance of Galois field inversion in GF(2^8), by so doing increasing security of AES against analytical attacks [106].

Diffusion Layer

This layer has two sublayers;

1. ShiftRow layer
   This layer utilizes cyclic shifting. The 16 input bytes are arranged into a four by four byte matrix. The second row is cyclic right shifted by three bytes, with the initial row left as it is, the third row is two byte right shifted. Lastly the fourth row is right shifted by one byte [27].

2. MixColumn layer
   A linear transformation is used by the MixColumn to enhance diffusion by combining all columns in the state matrix [90] and [27]. I denotes the 16 byte input states, and O denotes the output states, resulting in;

   $$MxColumn(I) = O$$

   The input to the MixColumn is the state obtained from the Output of the ShiftRow. This input matrix

   $$\begin{pmatrix} I_0 & I_4 & I_8 & I_{12} \\ I_5 & I_9 & I_{13} & I_1 \\ I_{10} & I_{14} & I_2 & I_6 \\ I_{15} & I_3 & I_7 & I_{11} \end{pmatrix}$$

is multiplied by a fixed four by four matrix

$$\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix}$$

with all the addition and multiplication done in GF(2^8) [27]. Recall that the input and output are 8-bit values representing an element in GF(2^8). The mathematical manipulation is computed in Galois field. The four by four matrix is represented in hexadecimal notation meaning that '01' in GF(2^8) polynomial coefficient form becomes 0000 0001.

Key Schedule

The sub-keys used in every round are derived from the original key [27]. The total number of keys required is the number of rounds increased by 1. The extra key is for key whitening in the initial key addition layer. For the key length of 128 bits, 11 sub-keys are required, 192-bit key length needs 13 sub-keys and 256-bit key length requires 15 sub-keys [27]. All sub-keys are of 128-bit each and are derived recursively. Key expansion arrays are used to store keys. These expansion arrays consist of words such as AES key schedule is word oriented [90]. Key schedules depends on the key length used as each has its own keys schedule. The process used to derive the sub keys in AES is termed Rijndael's key schedule [27].

Rijndael's Key Schedule for 128 Bit Key

44 words are required to accommodate the 11 sub-keys , as such expansion arrays are used to store them, these arrays range from W[0] to W[43]. The first four elements of the array contain the original 128-bit key, making up the first subkey k_0. The following sub-keys are computed recursively [27] and [90]. The first word of the following key is calculated as follows;

$$W[4z] = W[4(z-1)] + g(W[4z-1]) \tag{3.25}$$

Where g() is a non-linear function that inputs and outputs 4 byte, and z= 1.....10. The remaining 3 words of the sub-key are mathematically calculated recursively as follows;

$$W[4z+y] = W[4z+y-1] + W[4(z-1)+y] \tag{3.26}$$

where z ranges from 1 to 10 and j= 1,2,3. The input bytes in the non-linear function *g()* are rotated, byte-wise substitution is performed on them and finally RC (round coefficient) is added to it as shown in appendix C.1. This is done to eliminate symmetry in AES and add non-linearity. The round coefficient is an element of the Galois field GF(2^8) which only affects the most significant byte in the function. Every round has its own round coefficient which is the binary representation of that round i.e $RC[1] = x^0 = (00000001)_2$. The round coefficient of 128 bit key schedule is depicted in Appendix C.1

Rijndael's Key Schedule for 192 Bit Key

There are 52 words used as required by the 13 sub-keys. There are 8 iterations significantly less than the number of rounds. However, each iteration produces 6 words of the sub-key except the final iteration which produces 4 words. The first six words contain the original 192-bit key. The 6 words are computed as follows;

$$W[6z] = W[6(z-1)] + g(W[6z-1]) \tag{3.27}$$

where z ranges from 1 until 8. Recursively, the remaining 5 terms of the sub-key are computed as follows:

$$W[6z+y] = W[6z+y-1] + W[6(z-1)+y] \tag{3.28}$$

where z ranges from 1 to 8 and j= 1,2,3,4,5 except the final iteration where j=1,2,3.

The words are then divided into four words as each sub-key contains 128-bits. There are 8 round coefficients required by the non-linear functions g().

Rijndael's Key Schedule for 256 Bit Key

There are 60 words used as required by the 15 sub-keys. There are 7 iterations significantly less than the number of rounds. However, each iteration produces 8 words of the sub-key except the final iteration which produces 4 words. The first six words contain the original 192-bit key. With the following 6 words computed as follows;

$$W[8z] = W[8(z-1)] + g(W[8z-1]) \tag{3.29}$$

where z ranges from 1 until 7. The remaining 7 terms of the sub-keys are computed in a recursive manner as follows:

$$W[8z + y] = W[8z + y - 1] + W[8(z - 1) + y] \tag{3.30}$$

where z ranges from 1 until 7 excluding 5 and j= 1,2,3,4,6,7 except the final iteration where j=1,2,3. To determine the fifth word in every iteration an additional function is used that applies S-box to all four input bytes. The function is shown in appendix C.2. The fifth word is obtained as follows;

$$W[12z] = W[12(z - 1)] + h(W[12z - 1]) \tag{3.31}$$

During decryption all operations need to be inverted and the order of the round keys reversed.

3.3.1.4 Blowfish

The Feistel network is used as a basis for encrypting and decrypting the blowfish algorithm and consist of 16 rounds. Its key varies from 32 to 448 bits and encrypts block data of 64 bits [96]. Blowfish has two operations; key expansion and Data encryption:

- Key expansion: Initially the *p*-boxes and *s*-boxes are initialized using the hexadecimal representation of pi. There are eighteen *p*-boxes each containing 32 bits. Blowfish consists of four *s*-boxes each with 32 bits [96].

- Data encryption: The plaintext is broken down into two halves each consisting of 32 bits, R denotes the right half while L denotes the left side. A loop is then entered which will be repeated 16 times. L is then XORed with P_i which is the value of the *p*-box at the current index. R is XORed with a function of L. The function is a combination of four quarters. The *s*-box value of the first *s*-box is added to the value of the second *s*-box which is then XORed with the value of the third s-box and finally added to the value of the fourth *s*-box giving us the function f. finally L and R are swapped. This loop is repeated 16 times. L and R are then XORed with P_{17} and P_{18} respectfully. P and L are then combined to produce the ciphertext [102]. The structure of the blowfish algorithm is depicted in Figure 3.8 [102].

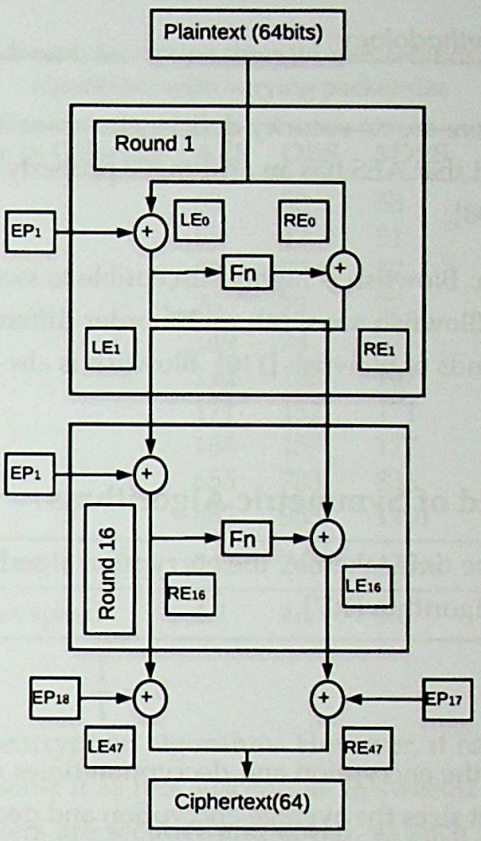


FIGURE 3.8: structure of the Blowfish algorithm

Data decryption in blowfish algorithm is the same as encryption with one difference; the *p*-boxes are used in reverse order.

3.3.2 Security of Symmetric Algorithms

The most important aspect of an encryption algorithm is its ability to resist attacks. The resilience of an encryption algorithm is judged on its key size. The greater the key size the stronger the encryption [108].

- DES: DES has a key size of 56-bit which can generate 7.2*1016 possible keys. With the advancement in processing power, DES has become susceptible to brute force attacks as it has a small key space [27].
- 3DES: 3DES has a key size of 168-bits, three times that of DES. It is highly susceptible to meet in the middle attacks [109]. Triple DES is theoretically vulnerable to linear attacks as well as differential attacks [110].

- AES: There are no security deficiencies associated with AES. However, it was observed that AES has an arithmetic property that might be susceptible to attacks [108].
- Blowfish: Blowfish is highly susceptible to sweet32 attacks [110]. Some of the keys in Blowfish are weak as 2^{nd} order differential attacks can be applied to four rounds of blowfish [110]. Blowfish is also vulnerable to reflection attacks [111].

3.3.3 Speed of Symmetric Algorithms

The ratio of the time taken by the encryption algorithm to execute is known as the speed of the algorithm [107].

$$S = \frac{1}{T} \tag{3.32}$$

By averaging the encryption and decryption times of the different algorithms with varying packet sizes the average encryption and decryption times of the algorithms can be obtained. Using a laptop with a 2.4GHz CPU, different file sizes ranging from 49 kb to 7.139MB were encrypted by Vashishtha et al [107] to obtain some performance metrics of symmetric algorithms among them being encryption and decryption times. their finding were used to populate tables 3.4 and 3.5 respectfully.

TABLE 3.4: Different encryption times(in milliseconds) of symmetric algorithms with varying packet size

Input size in (Kbytes)	AES	DES	3DES	Blowfish
49	56	29	54	36
59	38	33	48	36
100	90	49	81	37
247	112	47	111	45
321	164	82	167	45
694	210	144	226	46
899	258	240	299	64
963	208	250	283	66
5345.28	1237	1296	1466	122
7310.336	1360	1695	1786	107
Average time	374	389	452	60.3
Encryption speed	2.7	2.6	2.2	1.7

TABLE 3.5: Different decryption times(in milliseconds) of symmetric algorithms with varying packet size

Input size in (Kbytes)	AES	DES	3DES	Blowfish
49	63	50	53	38
59	58	42	51	26
100	60	57	57	52
247	76	72	77	66
321	149	74	87	92
694	142	120	147	89
899	171	152	171	102
963	164	157	177	80
5345.28	655	783	835	149
7310.336	882	953	1101	140
Average time	242	246	275.6	83.4
Decryption speed	4.1	4.1	3.6	12

Blowfish is the fastest encryption algorithm. However, it has several security vulnerabilities levelled against it as it is susceptible to sweet32 and reflection attacks. The two main parameters are security and speed, as such AES algorithm will be selected as it provides the highest level of security and is relatively fast during encryption and decryption. The main problem with algorithms utilizing the same key for both encryption and decryption is the delivery of the secret key as both the transmitter and receiver needs to have an identical key. The Diffie-Hellman key exchange [112] will be used to generate and distribute the keys among adjacent nodes.

3.4 Algorithm Formulation

This part explores the algorithm proposed to tackle security issues in OBS networks in relation to the control packet. Initially, it is assumed that all core and edge nodes are legitimate nodes and are equipped with three security measures;- SHA-256 to authenticate the integrity of the control packet, the Diffie-Hellman key exchange to create and distribute keys among nodes, and the AES algorithm for authenticating the confidentiality of the control packet.

The Diffie Hellman key exchange is used to establish the secret key used to encrypt and decrypt the control packet between nodes. This algorithm allows two parties to establish a shared secret key for them to communicate.

Adjecent nodes need to select key pairs, these key pairs can then be used to calculate a shared secret key to be used as the keying key to the AES encryption algorithm. To explain the key exchange process two nodes are considered denoted by N_a and N_b respectfully. The two nodes need to select key pairs, these key pairs are p and s. Where, p is a large prime number that is 1024 bits long so that the shared key is 128 bits long as required by AES. The second key pair s is the primitive root modulo of the prime number p. Once these two key pairs are chosen each node produces its own secret key r. Once all these keys are produced, the first node produce its own public key P as

$$P_{N_a} = s^{r_{N_a}} mod p \tag{3.33}$$

and the second node produces its public key in the same manner as

$$P_{N_b} = s^{r_{N_b}} mod p \tag{3.34}$$

The public key is then sent to the adjacent node in the network. Once the public key is received the nodes can then calculate the shared key between the two nodes. The first node calculates the shared key as

$$Sharedkey = P_{N_b}^{r_{N_a}} mod p \tag{3.35}$$

and the second node calculates the shared key as

$$Sharedkey = P_{N_a}^{r_{N_b}} mod p \tag{3.36}$$

The secret key computed by both nodes is identical and is used as the encryption key for the AES algorithm.

Once the control packet is generated, the initial transmitting node computes the message digest of the control packet. The control packet is padded until it is 64 bits shy of being a multiple of 512 bits. Modulo of the original message is used to compute the remaining 64 bits, which are then padded to the message to make it a multiple of 512 bits. Once padding is completed the message is divided into blocks of 512 bits. Each block is then subjected to 64 rounds of operation. The block is further broken down into 16 parts each containing 32 bits. These parts act as an input for the first 16 rounds. As for the remaining 48 rounds a word schedule W(i) is calculated and used as an input. The output of each block is fed as the input of the

following message block.

$$W(i) = W^{i-16} + \sigma^0 + W^{i-7} + \sigma^1 \tag{3.37}$$

Where,

$$\sigma^0 = (W^{i-15} ROTR^7(X)) \oplus (W^{i-15} ROTR^{18}(X)) \oplus (W^{i-15} SHR^3(X))$$
$$\sigma^1 = (W^{i-2} ROTR^{17}(X)) \oplus (W^{i-2} ROTR^{19}(X)) \oplus (W^{i-2} SHR^{10}(X))$$
$$ROTR^n(X) = \text{Circular right rotation of the input message } X \text{ by } n \text{ bits}$$
$$SHR^n(X) = \text{Circular right shift of the input message } X \text{ by } n \text{ bits}$$

Once all 64 schedule words are obtained the 8 hash variables are set to the initial chaining values which are;

Q_1 = 0x6A09E667, Q_2 = 0xBB67AE85,
Q_3 = 0x3C6EF372, Q_4 = 0xA54FF53A,
Q_5 = 0x510E527F, Q_6 = 0x9B05688C,
Q_7 = 0x1F83D9AB, Q_8 = 0x5BE0CD19.

After which the compression loop is carried out on all 64 schedule words. The compression function uses two temporary words which are used to update the hash variables. These words are computed as;

$$\begin{aligned} temp1 &= h + S1 + ch + k[i] + W[i] \\ temp2 &= S0 + maj \end{aligned} \tag{3.38}$$

Where,

$$ch = (Q_5 \wedge Q_6) \oplus (\neg Q_5) \wedge Q_7$$
$$maj = (Q_1 \wedge Q_2) \oplus (Q_1 \wedge Q_3) \oplus (Q_2 \wedge Q_3)$$
$$S0 = (Q_1 \ggg 2) \oplus (Q_1 \ggg 13) \oplus (Q_1 \ggg 22)$$
$$S1 = (Q_5 \ggg 6) \oplus (Q_5 \ggg 11) \oplus (Q_5 \ggg 25)$$

Once the temporary words are computed the schedule words are updated as follows;

Q_1 = temp1 + temp2 $Q_2 = Q_1$
$Q_3 = Q_2$ $Q_4 = Q_3$
$Q_5 = Q_4$ + temp1 $Q_6 = Q_5$
$Q_7 = Q_6$ $Q_8 = Q_7$.

After the compression function loop has been executed the hash values are updated

for the last time by adding the initial chaining values to the current hash values. Hash values are then concatenated to produce the hash value of the control packet. The hash value is then appended to the control packet and transmitted to the receiving node. Upon arrival the receiving node computes the hash value of the control packet of which is compared to the appended hash value. If the two match, then the control packet is processed for further action and if not, the control packet is dropped and the transmitting node is flagged throughout the network. Once flagged all traffic directed to this node is encrypted using AES-128.

The node transmitting the control packet to a compromised node initially breaks down the control packet into blocks of 128 bits. Blocks are further broken down into four-by-four columns consisting of sixteen bytes. The shared key computed through Diffie-Hellman between the two nodes is taken as the encryption key between the nodes. AES-128 requires 11 round keys and these keys are obtained from the shared key through the Rijndael's key schedule for 128 bits. key whitening is performed by combining the initial key with the data block, this is done to enhance the security of the block cipher.

Once key whitening has taken place the bytes are substituted in accordance with the lookup table presented in table 3.3. Bytes are substituted to introduce confusion into the blocks as the bytes are altered in a non linear fashion. After byte substitution the data moves to the shift rows. Shift rows shift the second row cyclically by three bytes to the right, while the third and fourth rows are right cyclic shifted by two and one byte respectfully. The first row is left unaltered. Shift rows introduce diffusion into the data as it is shifted from its original position hence helping to obscure the control packets data. From there more diffusion is introduced by the MixColumn layer where the four by four columns obtained after the shift rows are multiplied with a fixed four by four matrix shown in equation 2. MixColumn introduces diffusion by vertically altering the data while ShiftRows alters it horizontally. After the MixColumn layer the second round key is added to the data to add greater confusion. This process is repeated nine times with the data passing through all the four steps: Byte substitution, ShiftRows, MixColumns and round key addition, with a different round key added every round. The final tenth round does not include the Mixcolumn layer as it would just be consuming unnecessary processing power without changing the data in any form.

Once the encrypted packet arrives at the destination node, the receiving node decrypts the control packet by performing AES encryption in reverse and computes

the hash value to check if confidentiality has been violated. If the hashes do not match then the control packet is dropped, if the hashes match then the control packet is received and processed for further action. This process is repeated until the control packet reaches its intended edge node. The algorithm is fully illustrated in algorithm 5.

Algorithm 5 Secure burst control packet mechanism

1: **Notation;**
2: N[i]:n^{th} node.C_p:Control packet.D_b:Data burst. $E_n_C_p$: Encrypted control packet. $D_c_C_p$: Decrypted control packet. M_d: message digest. k_{prv}: private key. $M_d_C_p$:appended message digest to control packet;
3: Create N[i].k_{prv};
4: Broadcast N[i].k_{prv} ta adjacent nodes;
5: Recieve and save N[i].k_{prv};
6: **while** Not end of traffic **do**
7: Create C_p and D_b;
8: Compute M_d of C_p;
9: **if** N[i+1] is flagged **then**
10: Encrypt C_p;
11: $E_n_C_p$= C_p(N[i].k_{prv});
12: Send $E_n_C_p$ to N[i+1];
13: Reciever decrypts C_p;
14: $D_c_C_p$= $E_n_C_p$(N[i].k_{prv});
15: Receiver computes M_d of C_p;
16: **if** computed M_d==Appended M_d **then**
17: Reciever processes C_p;
18: **if** N[i+1] != Destination **then**
19: Jump to 9;
20: **else**
21: End;
22: **else**
23: Receiver drops C_p;
24: Node N[i] is flagged;
25: Request retransmission;
26: **else**
27: Send $M_d_C_p$ to N[i+1];
28: Receiver computes M_d of C_p;
29: **if** computed M_d==Appended M_d **then**
30: Receiver processes C_p;
31: **if** N[i+1] != Destination **then**
32: Jump to 9;
33: **else**
34: End;
35: **else**
36: Receiver drops C_p;
37: Node N[i] is flagged;
38: Request retransmission;
39: **End;**

3.5 Conclusion

In this chapter an algorithm to tackle security threats aimed at the control packet in OBS networks is developed. The algorithm should be able to validation the integrity of the control packet between nodes. Once the integrity of the control packet is violated the algorithm should also be able to apply security protocols at compromised nodes to ensure confidentiality of the control packet. An in-depth analysis of message authentication codes focused on guaranteeing integrity was explored. Speed and security were the main metrics being considered when exploring these message authentication codes as such, SHA-256 was chosen as it provided the right blend of security and speed. The most prevalent method of securing data is the use of cryptography. Cryptographic techniques were analyzed to determine a technique that was both fast and secure. Since, symmetric algorithms are faster than asymmetric algorithm focus was given to symmetric algorithms. Blowfish is the fastest of the explored symmetric algorithms. However, it has a number of security vulnerabilities. AES does not have any security vulnerabilities and is relatively fast, as such AES-128 was chosen to preserve the confidentiality of data. The solution is formulated based on the AES for enforcing the confidentiality of the control packet and SHA-256 algorithms for validating the integrity of the control packet through per hop authentication. The next chapter presents the validation technique used to evaluate the proposed algorithm.

Chapter 4

Results Analysis

4.1 Introduction

There are a variety of security metrics that can be evaluated, however since security and the fast deliverance of data to its intended destination is our main concern, the metrics under consideration are end-to-end delay, throughput and burst loss ratio. This chapter presents performance evaluation of the proposed algorithm employing SHA-256 for per hop authentication of the control packet and AES-128 for preserving the confidentiality of the control packet. The proposed algorithm is compared to the conventional AES-128, RSA algorithm and a network with no security applied to it.

4.2 Evaluation Framework

The algorithms are evaluated using the OMNET++ simulator. OMNET was chosen because it is an open source simulator, its ease of learning, its ability to integrate new modules into the simulator and its ease to manipulate existing modules.

OMNET++ is a protrusile, modular, component-based C++ simulation library and framework used for constructing network simulators [113]. OMNET++ supports the building of both wireless and wired networks, queuing networks and on chip networks. The simulator also supports development of independent projects to produce desired functionalities of modules such as support for wireless ad-hoc networks, optical networks, sensor networks, photonic networks, internet protocols and performance modelling. The OMNET++ framework has a number of tools including but not limited to a runtime environment and an IDE [113]. It also supports a whole host of other functionalities such as integrating a database, emulating a network, SystemC integration and simulating in real time [113].

INET framework which is an open source library model for OMNET++ which provides models, protocols and agents for communication networks. INET has a variety of internet stack models varying from Internet Protocol versions 4 and 6 to Open Shortest Path First. INET also contains multiple link layer protocols for both wireless and wired networks along with other components and protocols [113]. OBS modules and other modules are built ontop of INET.

OBS Modules for OMNET++

To simulate OBS networks in OMNET++ OBS-modules are used. A triad of OBS network architectures exist. These architectures are: Emulated OBS (E-OBS), Conventional OBS (C-OBS), and Labelled OBS (L-OBS). These architectures are differentiated by how the manner in which the transmission and control packet manipulation is carried out. The models are based on a C-OBS architecture. The offset time needs to allow enough time for the processing of the control packet and configuration of the switching matrix as it is fixed at each node.

Mechanisms, techniques and functionalities of the modules are implemented using C + +, while the architecture of the network is implemented using the NED language. The OBS architecture consists of edge nodes and core nodes [114].

OBS Network Architecture

There are two type of nodes that make up an Optical burst switched network: the edge node and the core node which are connected by optical fibres. These optical fibres constitute several data and control channels.

Edge Node

There are two types of edge nodes; ingress edge nodes and egress edge nodes. TCP/IP packets are aggregated into bursts in the ingress node based on the destination and quality of service criteria. The burst is then sent into the network. Once the burst reaches the egress node, it is disassembled into packets. The edge node consists of a dispatcher, assembler, Scheduler, control packet generator as depicted in Figure 4.1 below.

- A dispatcher classifies IP packets based on predefined criteria and sends packets of the same class to the same assembler.

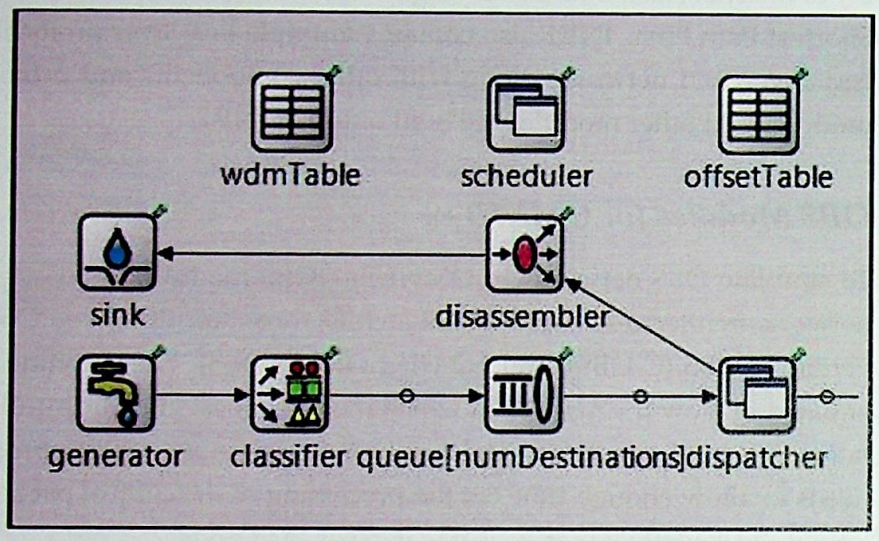


FIGURE 4.1: Edge node architecture

- The assembler is responsible for burst creation by aggregating IP packets into bursts. Burst creation can be achieved by setting the size of the burst and transmitting the burst once the size is reached. Burst creation can also be achieved by setting a time and once the timer runs out the burst is sent into the network.
- The scheduler selects a queue to serve and triggers the control packet generator by a signal.
- The control packet generator selects the output port, wavelength and the time the burst is sent into the network based on the information received from the scheduler.

Core Node

The core node has three major functions; processing the control packet, routing the burst and the control packet, and finally reserving resources for the control packet and the burst. These functionalities are achieved by the control unit and the optical switch fabric depicted by the architecture of the core node is shown in Figure 4.2.

- Control Unit: The main task of the control unit is to process the control packet. The control unit has dedicated channels for the control packets. Once the control packet reaches the control unit, it is converted to electronic format and the information contained within it is retrieved. The information is then used to update the routing table and to reserve resources for the incoming burst.
- Signaling and routing unit: This unit uses the information obtained by the control unit to create, maintain, and update the routing table.

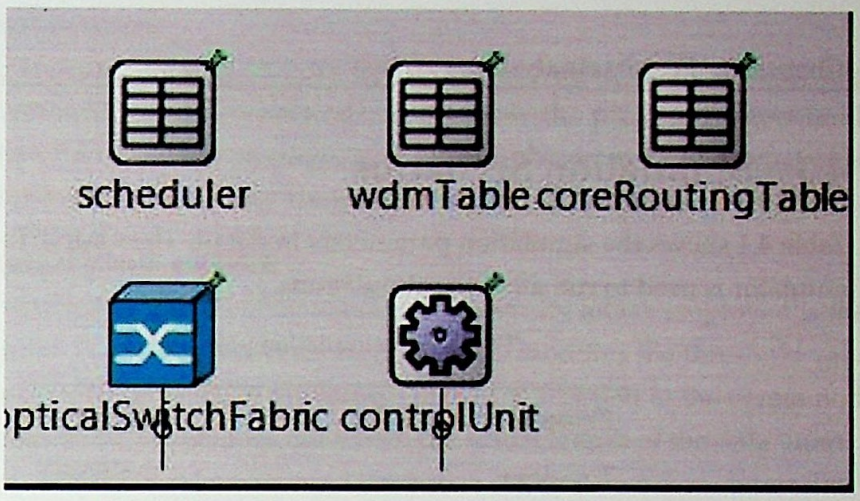


FIGURE 4.2: core node architecture

Signaling in OBS Modules for OMNET++

A signaling scheme is used to configure switches and allocate resources needed by the burst at each node. Signaling techniques can be classified by the manner in which they reserve resources, the way they deal with contention, how they release the resources after a burst is switched through a node and the manner in which they reserve resources. The signaling techniques implemented are;

- Just In Time(JIT):Reservation of resources begins after the offset time contained in the control packet
- Just Enough Time (JET): Reservation of resources begins once the initial bit of the burst reaches the core node and is released after the last bit is transmitted.

Scheduling in OBS Modules for OMNET++

The main purpose of the scheduler is to assign wavelengths to bursts. Wavelength is chosen depending on the size of the burst as it needs to be able to accommodate it. The LAUC (Latest-Available- Unscheduled Channel) or HORIZON algorithm is used. This LAUC algorithm selects an available wavelength that has the nearest and reserved time for the burst.

The following section looks at scenarios explored to assess the integrity of the formulated algorithm as compared to other existing algorithms.

4.3 Simulation Scenarios

Table 4.1 shows the simulation parameters in detail. The OMNET++ simulator and emulator is used to run all of the simulations.

TABLE 4.1: simulation parameters

Parameter	**levels**
Topology	NSFNET
Bandwidth per wavelength(Gbps)	10
Number of wavelengths	4
Control packet length	1250bytes
Scheduling algorithm	LAUC
Signaling technique	JIT

Algorithms are implemented on the National Science Foundation Network (NSFNET) topology depicted in Figure 4.3. metrics being assessed are throughput, burst loss ratio, and end-to-end delay. The proposed algorithm is compared to the conventional AES-128 and RSA algorithms applied to the core nodes. All the 3 algorithms are further compared to a network operating with no security measures employed.

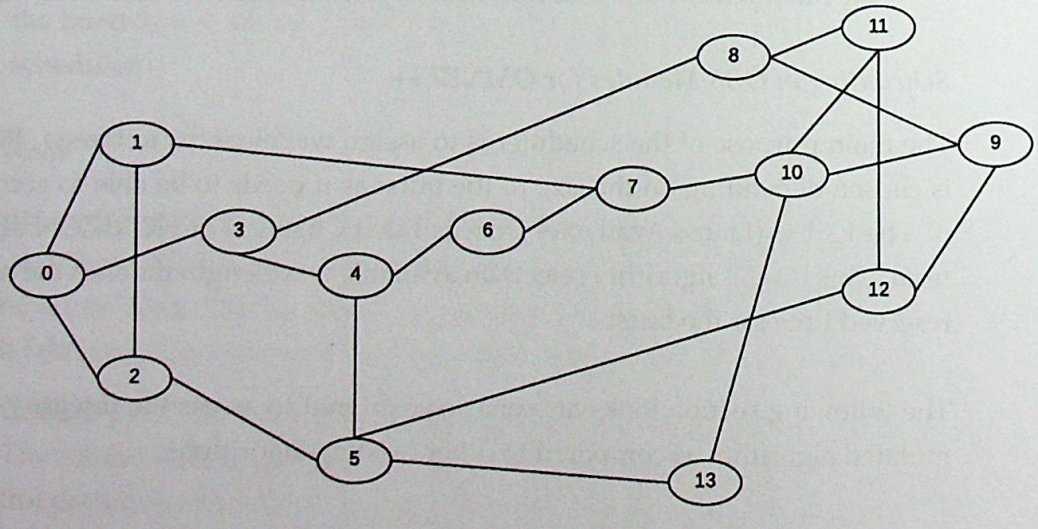


FIGURE 4.3: simulation topology

To evaluate the proposed mechanism two scenarios were investigated and these scenarios are;

- **Ideal network**
  A network under no security threats incorporating all the security algorithms mentioned. This is done to determine how the proposed algorithm behaves when there are no compromised nodes in the network and what effect it has on the end-to-end delay, throughput and burst loss ratio.

- **Compromised network**
  A network under security attacks. The security attack employed is the burstification attack where a compromised node modifies the threshold value in the control packet, as such the control packet is dropped at the egress node as the stored burst length does not match the actual length of the data burst. Initially five compromised nodes are introduced in the network to determine their effects on the performance metrics being investigated. Five more compromised nodes are added to the network to determine how the formulated algorithm handles an increase in security threats.

The simulation is based on the 14 node NSFENT topology shown in Figure 4.4 below. The simulated scenarios uses five edge nodes. Each edge node is used as a source and destination node, hence, the network utilizes five sources and five destinations as depicted in Figure 4.4 showing the simulation scenario of the network.

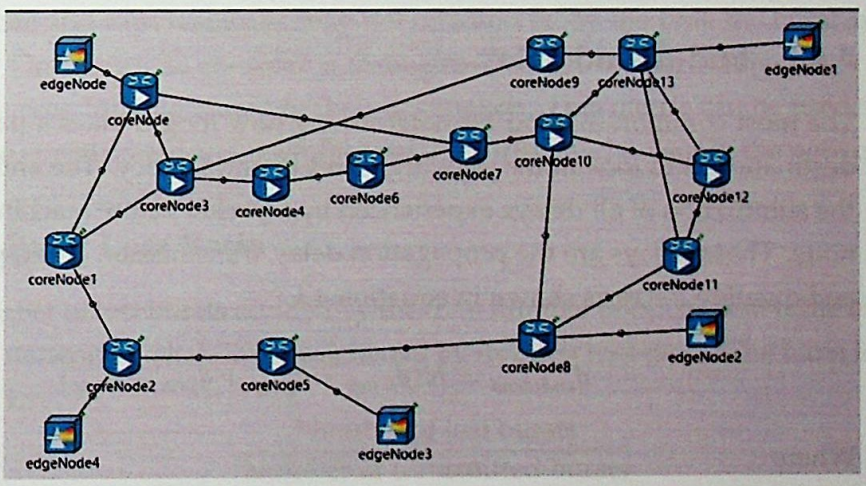


FIGURE 4.4: simulation scenario

Every edge node has four destinations as it sends data to all other edge nodes. Network traffic is defined by the number of packets produced by every edge node that are varied by 100 packets from 100 up to 1000 packets. Initially all core nodes are deemed to be legitimate nodes and no security mechanisms are implemented in

them. Secondly core nodes are equipped with RSA encryption and decryption algorithm to ensure the confidentiality of the control packet throughout the network. Regardless of whether the node is compromised or not, once the control packet reaches the node it is decrypted, processed and then encrypted before being transmitted out of the node. Lastly all the nodes are equipped with sha-256 authentication algorithm and AES encryption and decryption algorithm. In the first instance all nodes are deemed to be legitimate nodes and so control packet authentication is carried out as the control packet travels throughout the network. Compromised core nodes are then introduced into the network. This is done to determine the effect on the end-to-end delay, throughput, burst loss ratio in network.

The next part looks at performance metrics used to assess the proposed algorithm. The end-to-end delay, throughput, and burst loss ratio are the performance metrics in question.

4.4 Performance Metrics

4.4.1 End-to-end delay

The most common method for determining how long it takes a packet to reach its destination is to look at the network's end-to-end latency. The end-to-end delay is the summation of all delays experienced by a packet before reaching its destination node. These delays are the propagation delay, transmission delay, processing delay and queuing delay as shown in equation 4.1.

$$d_{endtoend} = N(d_{trans} + d_{prop} + d_{proc} + d_{que}) \tag{4.1}$$

Where

- $d_{endtoend}$ is the end-to-end delay
- d_{trans} is the transmission delay
- d_{prop} is the propagation delay
- d_{proc} is the time required by the node to process the control packet
- d_{que} is the queuing delay
- N is the number of nodes

The transmission delay is the required time for a node to thrust all the packets bits into the link. A signal takes time to travel from source to destination through the transmission medium being used, this then gives rise to the propagation delay which is proportional to the distance traversed by the packet. Once the packet reaches a node in the network it needs to be serviced to determine its optimum path and reserve network resources, the time required to service this packet at the node gives rise to whats known as the processing delay. A node can process one control packet at a time. If the control packets are received at a faster rate than the node can process them then the packets are stored in a queue until they can be processed and transmitted thus increasing the queuing delay.

The end-to-end delay is used to determine how long a packet takes from source to destination as such is an ideal metric to determine the availability of data in a network.

4.4.2 Throughput

Security is very important in network communication. However, sometimes the implementation of security protocols lead to a reduction in the successful delivery of packets as the increase in network delay introduced by the protocols reduces the offset time and thus resources may not be reserved by the time the burst arrives at a node. One metric to consider is throughput which measures how many packets where successfully delivered to their destination. This metric can be used to determine the effectiveness and usefulness of a security mechanism in the network.

4.4.3 Burst Loss Ratio

The number of lost bursts divided by the total number of bursts sent is the burst loss ratio. A network's reliability is calculated by its burst loss ratio.. The burst loss ratio is given by;

$$\frac{\text{Number of lost bursts}}{\text{Number of transmitted bursts}} \tag{4.2}$$

Simulation results obtained from the OMNET++ simulation framework pertaining to the metrics explained above are presented in the subsequent section. Initially throughput is explored followed by the burst loss ratio and finally the end-to-end delay.

4.5 Experimental Results

4.5.1 Throughput

Figure 4.5 depicts the findings, which indicate that the throughput is highest when there are no security measures applied to the core nodes. This is because the application of security mechanisms increase the processing delay at those nodes hence reducing the time difference between the burst and the control packet leading to some bursts being dropped as no resources are allocated to it at that node. The processing speed of the implemented security mechanism is directly proportional to the dropped bursts. The proposed algorithm does not drop as many bursts as the RSA algorithm however drops more bursts as compared to the AES algorithm. This suggests that the AES algorithm is significantly quicker than the proposed algorithm in terms of processing speed. However, the proposed algorithm and the AES algorithm are both much faster than the RSA algorithm in terms of processing speed.

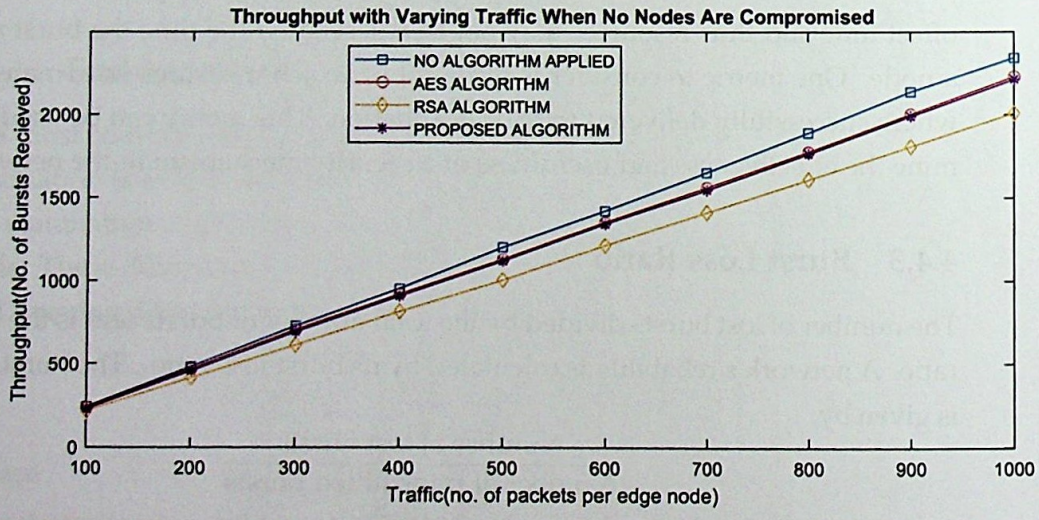


FIGURE 4.5: received bursts with varying traffic for normal scenario

When compromised nodes are introduced into the network, the throughput of RSA and AES implementation remains the same as depicted in Figure 4.6 and 4.7 when five and ten nodes are compromised respectfully. This is due to the fact that these algorithms are implemented at all core nodes regardless of a presence of a compromised node. As such, the presence of a compromised node does not affect the throughput of the network. The throughput of the proposed algorithm decreases as compromised nodes are detected in the network. This is due to the fact that once a node is compromised and subsequently changes any field in the control packet, the

control packet will fail authentication at the next node and hence will be dropped from the network along with its corresponding burst. All the control packets from that compromised node will be dropped until all traffic to that compromised node is encrypted hence reducing the successful delivery of the control packets along with their successive bursts. As such when compromised nodes increase in the network the throughput decrease as shown by the discrepancy of the throughput when there are five compromised nodes and when there are ten compromised nodes.

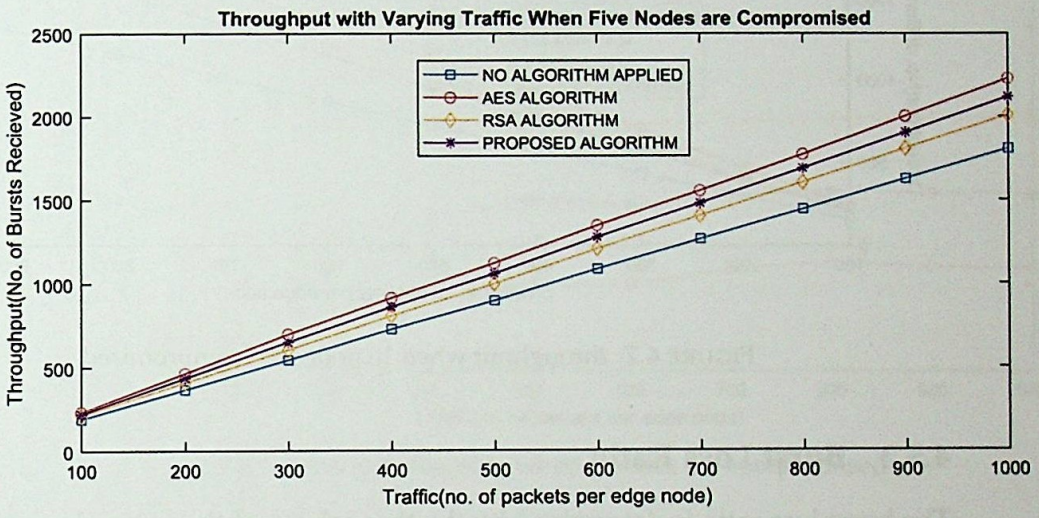


FIGURE 4.6: throughput when 5 nodes are compromised

As the number of compromised nodes in the network increases, the proposed algorithm's throughput decreases. The throughput of RSA and AES algorithms is constant regardless of how many compromised nodes there are in the network. As compromised nodes increase and there are no security measures taken on the network there is a drastic decrease in throughput as bursts are not delivered to their intended destinations. As such the proposed algorithm is viable as it increases network throughput and has a higher throughput than that of RSA.

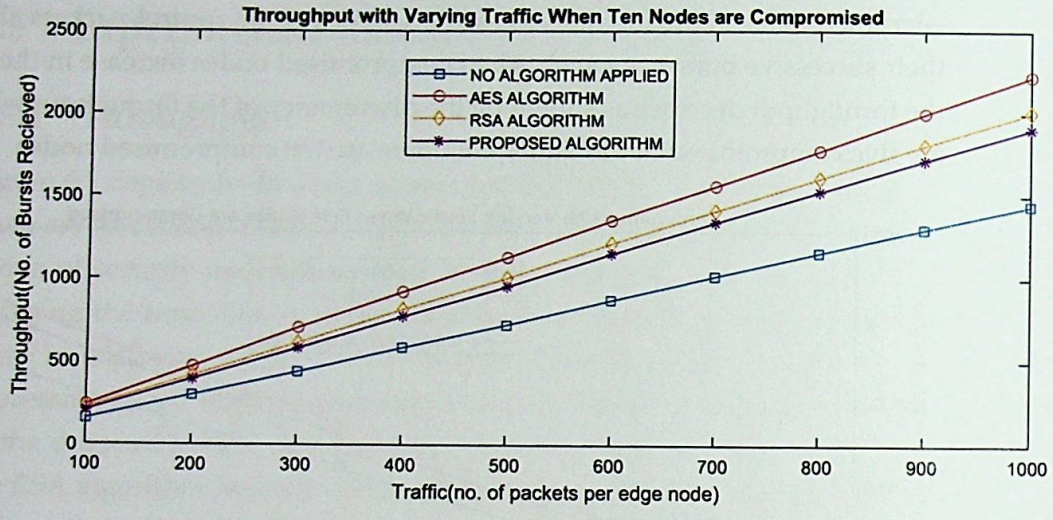


FIGURE 4.7: throughput when 10 nodes are compromised

4.5.2 Burst Loss Ratio

The burst loss ratio is determined by the throughput of the network as such has a direct correlation to the throughput. Figure 4.8 shows the burst loss ratio as traffic is varied when a network has no security threats and no security mechanisms are applied.

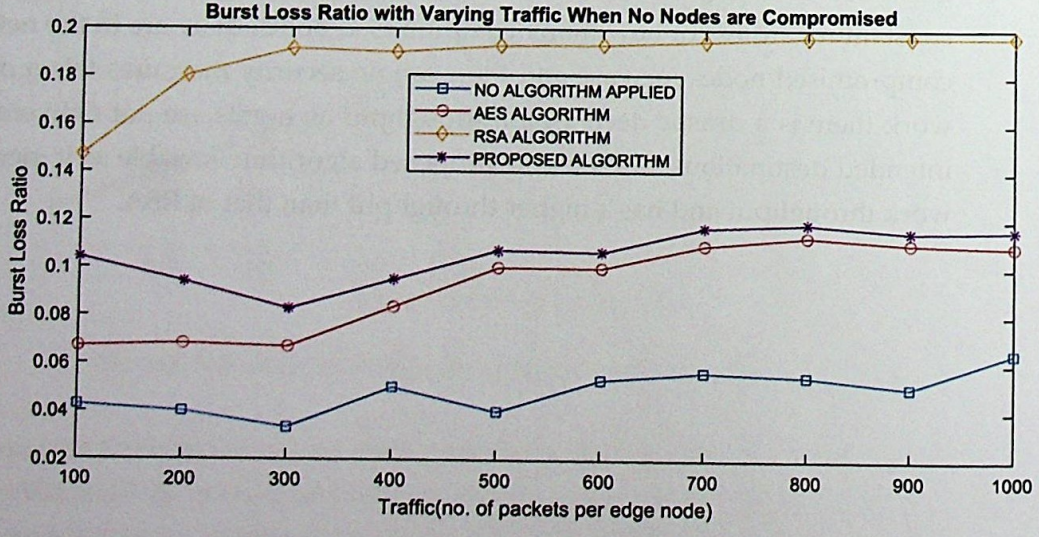


FIGURE 4.8: burst loss ratio for normal scenario

As the number of compromised nodes increases, so does the burst loss ratio of the proposed algorithm. The burst loss ratio is indirectly proportional to the throughput. The throughput is indirectly proportional to the number of nodes that have

been compromised. As compromised nodes increase the burst loss ratio also increase. Figure 4.8 and 4.9 show the discrepancy of the burst loss ratio as compromised nodes are introduced into the network. The burst loss ratio of AES and RSA algorithm are constant throughout all the simulation scenarios as their throughput are also constant.

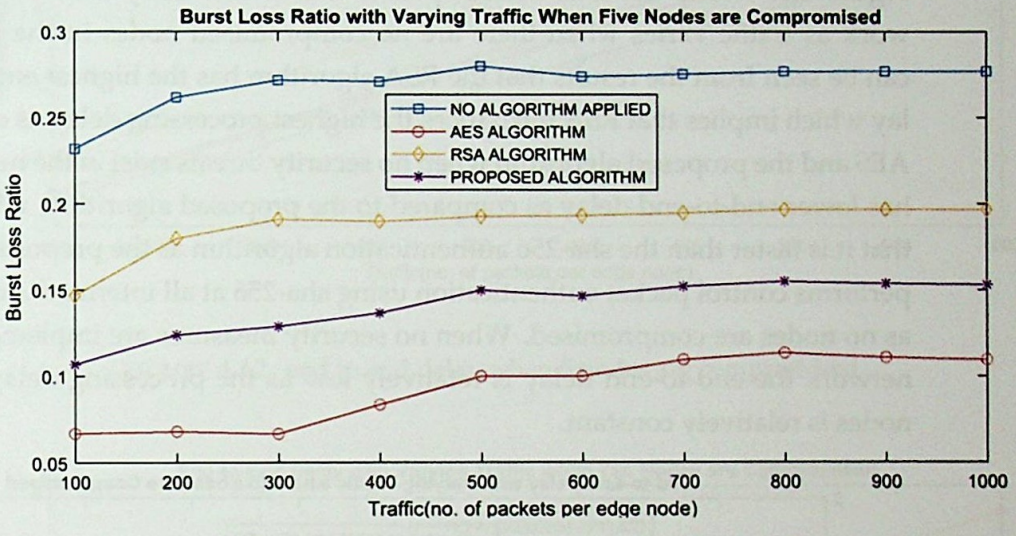


FIGURE 4.9: burst loss ratio when 5 nodes are compromised

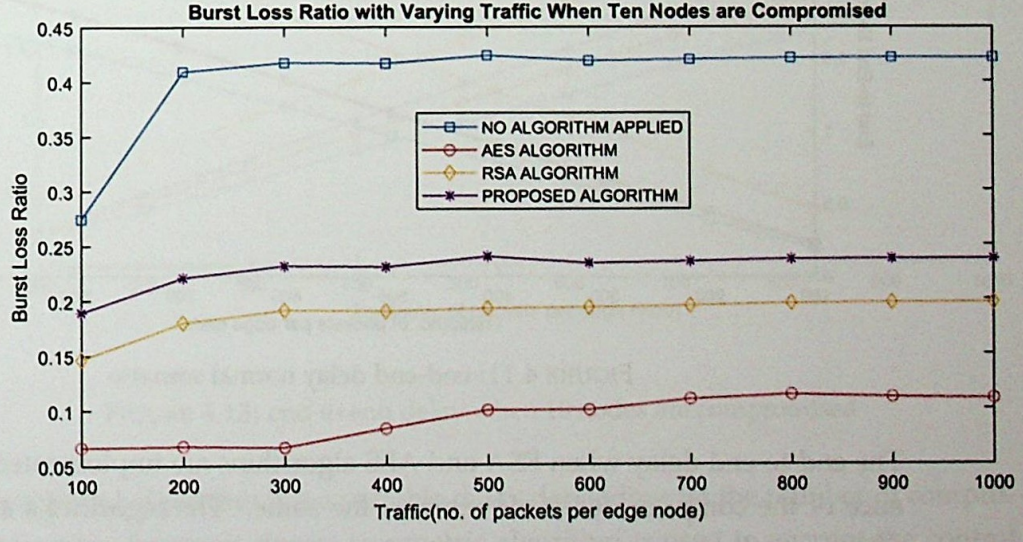


FIGURE 4.10: burst loss ratio when 10 nodes are compromised

4.5.3 End-to-End Delay

Figure 4.11 shows the total control packet end-to-end delay incurred by the network as traffic varies when there are no compromised nodes in the network. It can be seen from the results that the RSA algorithm has the highest end-to-end delay which implies that RSA introduces the highest processing delay as compared to AES and the proposed algorithm when no security threats exist in the network. AES has lower end-to-end delay as compared to the proposed algorithm, which implies that it is faster than the sha-256 authentication algorithm as the proposed algorithm performs control packet authentication using sha-256 at all intermediate core nodes as no nodes are compromised. When no security measures are implemented in the network the end-to-end delay is relatively low as the processing delay at all core nodes is relatively constant.

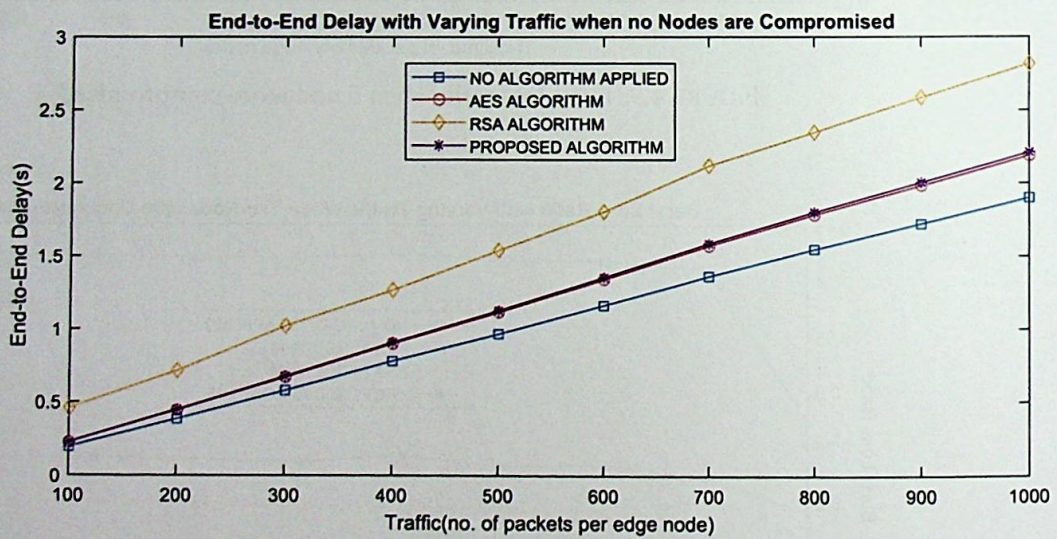


FIGURE 4.11: end-end delay normal scenario

The end-to end delay when RSA and AES algorithms are implemented in the presence of the compromised node remains the same. The algorithms are always in effect regardless of a compromised node as shown by the constant delay observed in Figure 4.12 and Figure 4.13. However, the end-to-end delay of the proposed algorithm increase due to the fact that the control packet has to be encrypted before being transmitted to the compromised node, decrypted by the compromised node, processed, encrypted before being transmitted out of the compromised node and finally decrypted by the successive node in the network. This all increases the processing delays at the nodes and hence increases the end-to-end delay.

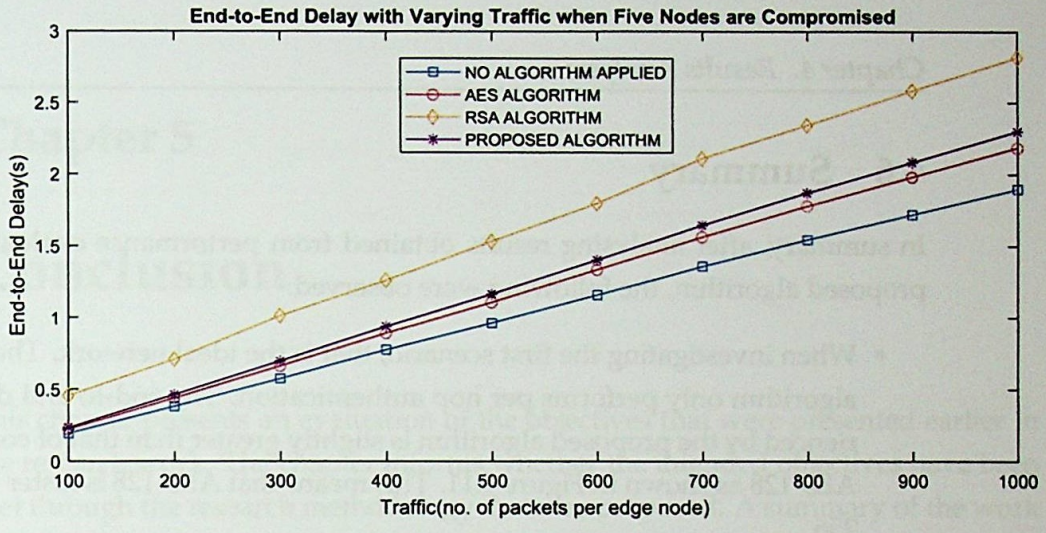


FIGURE 4.12: end to end delay when 5 nodes are compromised

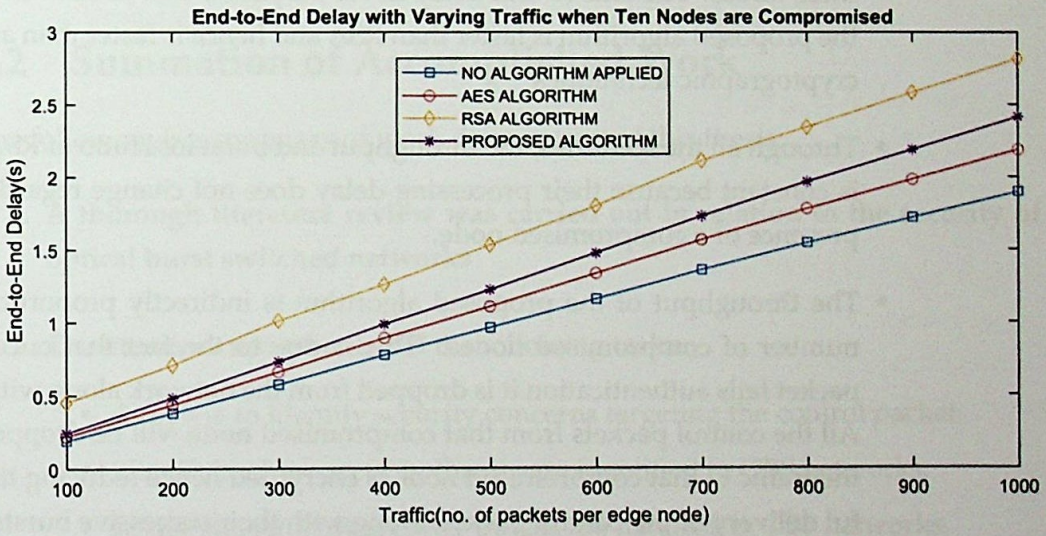


FIGURE 4.13: end to end delay when 10 nodes are compromised

The proposed algorithm has a variable delay depending on the number of compromised nodes, however since a symmetric algorithm is used to encrypt the control packet the delay will never surpass that of RSA as symmetric algorithms are significantly faster than asymmetric algorithms.

4.6 Summary

In summary, after analysing results obtained from performance evaluation of the proposed algorithm, the following were observed:

- When investigating the first scenario, that is the ideal network. The proposed algorithm only performs per hop authentication. The end-to-end delay experienced by the proposed algorithm is slightly greater than that of conventional AES-128 as shown in Figure 4.11. This means that AES-128 is faster than SHA-256.

- The proposed algorithm's end-to-end delay is proportional to the number of compromised nodes, as a compromised node increases the processing delay at three nodes. The end-to-end delay never surpasses that of RSA showing that the proposed algorithm is faster than RSA and hence is faster than asymmetric cryptographic techniques.

- Through all the scenarios, the throughput and burst loss ratio of RSA and AES is constant because their processing delay does not change regardless of the presence of a compromised node.

- The throughput of the proposed algorithm is indirectly proportional to the number of compromised nodes. This is due to the fact that once a control packet fails authentication it is dropped from the network along with its burst. All the control packets from that compromised node will be dropped until all the traffic to that compromised node is encrypted hence reducing the successful delivery of the control packets along with their successive bursts.

- The throughput of RSA and AES is constant because compromised nodes do not affect the performance of the algorithms in any way.

- Network resources are always available as compromised nodes are used in the network.

Chapter 5

Conclusion

This chapter presents an evaluation of the objectives that were presented earlier in the research work. The chapter informs whether the intended objectives have been met through the research methodology that was proposed. A summary of the work carried out is also presented in this chapter.

5.1 Conclusion

5.2 Summation of Accomplished Work

The following is a summary of work done as part of this thesis:

1. **A thorough literature review was carried out in relation to the security of optical burst switched networks**

   The literature review entailed the following tasks:

   - A review to identify security concerns targeting the control packet.
   - Identifying how security threats are identified in OBS networks.
   - To distinguish how security threats are handled in OBS networks.

   It was observed that most security threats in OBS networks are levelled against the control packet. Existing security mechanisms used to counteract security threats in OBS networks are of two types: nodal isolation and the use of cryptographic methods to protect the confidentiality of data. The isolation of a node from the network results in low network resource utilization while the application of cryptographic techniques increases the time required for the control packet to travel through the network. Studies graphical differentiation of existing mechanisms is given in the table 5.1 below.

TABLE 5.1: Exisitng OBS security mechanisms

Security mechanisms in OBS	**Studies**
cryptographic methods	[57], [73], [59], [54]
Nodal isolation	[51], [61], [52], [62], [46], [63], [60].

2. **Propose a mechanism that maximizes network resource utilization without increasing the end to end delay**

   To design a secure mechanism that reduces the end-to-end delay in a network while ensuring the maximum availability of nodes in the network: an analysis of message authentication codes was carried out to ascertain the best method to validate the integrity of the control packet, as well as an analysis of cryptographic techniques to determine the best method to guarantee the confidentiality of the control packet through the network. The algorithm is formulated based on the AES-128 for enforcing the confidentiality of the control packet and SHA-256 algorithms for validating the integrity of the control packet through per hop authentication as discussed in section 3.4.

3. **Simulation Validation**

   The algorithm was validated using the OMNET++ simulation network utilizing two simulation scenarios. These scenarios being: a network under no security threats and a network under security attack. The algorithms under observation were the proposed algorithm, AES-128, RSA and a network with no security algorithm applied. The metrics under consideration where the end-to-end delay, throughput and burst loss ratio.

   The observed results showed that the proposed algorithm maximized network resource utilization as all nodes were always available. However, the proposed algorithm has a lower end-to-end delay in relation to asymmetric cryptographic algorithms and higher delay than symmetric cryptographic techniques as shown by Table 5.2.

TABLE 5.2: Comparison of simulated algorithm's based on throughput, end-to-end delay and burst loss ratio

ALGORITHMS	Throughput	End-to-end delay
AES	High	Low
RSA	Low	Low
Proposed Algorithm	Medium	Medium

In conclusion, the algorithm proposed in this research provides a secure mechanism for securing a control packet in OBS networks. This mechanism uses per hop authentication utilizing SHA-256 to determine a compromised node. All compromised nodes were identified in the network. However, 2%-5% of the bursts were dropped from the network while identification was being carried out depending on the number of traffic in the network. After a compromised node has been identified AES-128 is used to ensure the confidentiality of the control packet as it is being transmitted to a compromised nodes. The inclusion of SHA-256 and AES-128 increases the processing delay at the nodes which in-turn increases the end-to-end delay in the network. The proposed algorithm can be used as an alternative to methods employing nodal isolation techniques as it continues to utilize a node after it has been deemed compromised ensuring that resources are always available in the network. The proposed algorithm may not be a viable alternative to symmetric cryptographic methods as they provide higher throughput and lower end-to-end delay due to their fast processing speed which means a lower processing delay at nodes and a smaller offset time. The proposed algorithm can be used in lieu of asymmetric cryptographic methods as it provides a lower end-to-end delay and higher throughput as asymmetric cryptographic methods have longer processing delay.

Appendix A

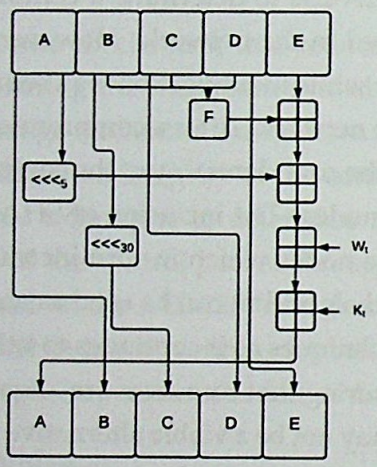


FIGURE A.1: compression function of sha-1

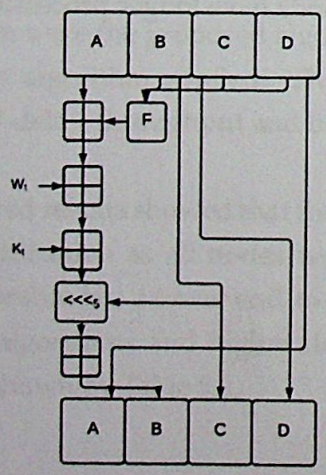


FIGURE A.2: compression function of MD5 algorithm

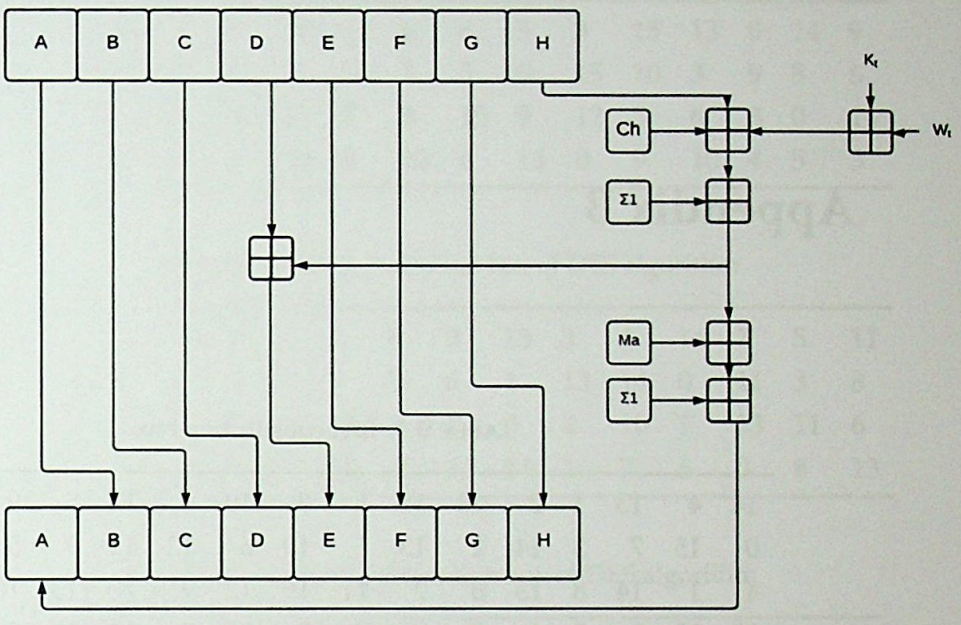


FIGURE A.3: compression function of sha256 algorithm

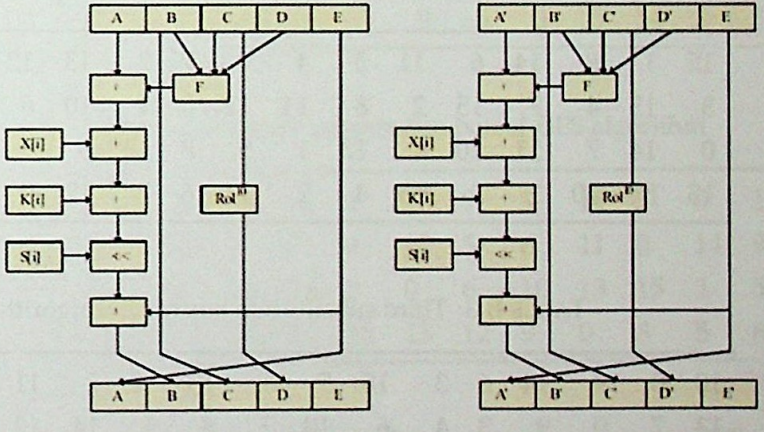


FIGURE A.4: compression function of RIPEMD-160 algorithm

Appendix B

TABLE B.1: first substitution box

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	3	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	13	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

TABLE B.2: second substitution box of DES algorithm

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	5	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

TABLE B.3: Third substitution box of DES algorithm

10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

TABLE B.4: Fourth substitution box of DES algorithm

7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

TABLE B.5: fifth substitution box of DES algorithm

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

TABLE B.6: Sixth substitution box of DES algorithm

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

TABLE B.7: Seventh substitution box of DES algorithm

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

TABLE B.8: Eighth substitution box of DES algorithm

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

TABLE B.9: lookup table for Expansion permutation in DES

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

TABLE B.10: Permutation table in DES

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

Appendix C

TABLE C.1: Inverse multiplicative table used within the AES S-Box obtained from [27]

		Y														
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	0	00	01	8D	F6	CB	52	7B	D1	E8	4F	29	C0	B0	E1	E5	C7
	1	74	B4	AA	4B	99	2B	60	5F	58	3F	FD	CC	FF	40	EE	B2
	2	3A	6E	5A	F1	55	4D	A8	C9	C1	0A	98	15	30	44	A2	C2
	3	2C	45	92	6C	F3	39	66	42	F2	35	20	6F	77	BB	59	19
	4	1D	FE	37	67	2D	31	F5	69	A7	64	AB	13	54	25	E9	09
	5	ED	5C	05	CA	4C	24	87	BF	18	3E	22	F0	51	EC	61	17
	6	16	5E	AF	D3	49	A6	36	43	F4	47	91	DF	33	93	21	3B
	7	79	B7	97	85	10	B5	BA	3C	B6	70	D0	06	A1	FA	81	82
X	8	83	7E	7F	80	96	73	BE	56	9B	9E	95	D9	F7	02	B9	A4
	9	DE	6A	32	6D	D8	8A	84	72	2A	14	9F	88	F9	DC	89	9A
	A	FB	7C	2E	C3	8F	B8	65	48	26	C8	12	4A	CE	E7	D2	62
	B	0C	E0	1F	EF	11	75	78	71	A5	8E	76	3D	BD	BC	86	57
	C	0B	28	2F	A3	DA	D4	E4	0F	A9	27	53	04	1B	FC	AC	E6
	D	7A	07	AE	63	C5	DB	E2	EA	94	8B	C4	D5	9D	F8	90	6B
	E	B1	0D	D6	EB	C6	0E	CF	AD	08	4E	D7	E3	5D	50	1E	B3
	F	5B	23	38	34	68	46	03	8C	DD	9C	7D	A0	CD	1A	41	1C

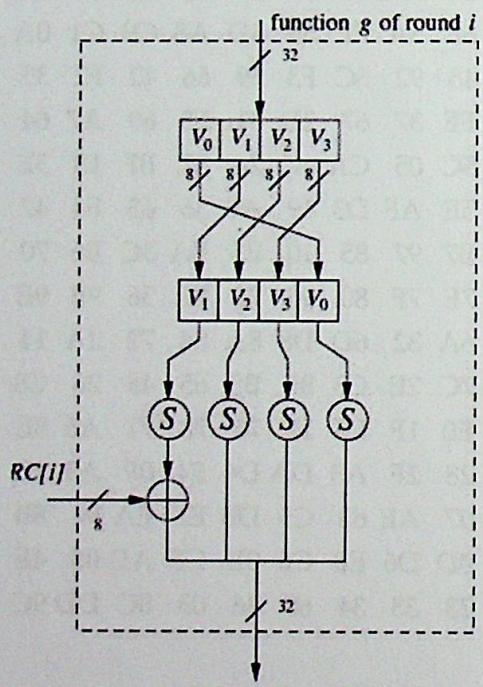


FIGURE C.1: Round coefficient of 128 bit key schedule

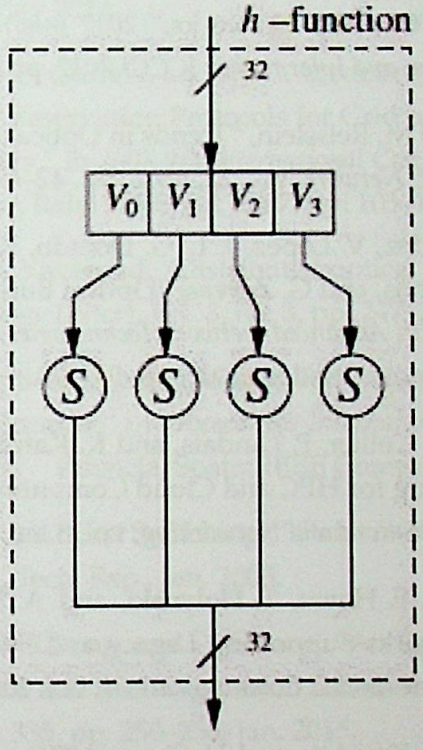


FIGURE C.2: S-box function in 256-bit key schedule

Bibliography

[1] N. Sreenath, K. Muthuraj, and G. V. Kuzhandaivelu, "Threats and Vulnerabilities on TCP/OBS Networks," *2012 International Conference on Computer Communication and Informatics, ICCCI 2012*, pp. 1–5, 2012.

[2] M. Maier and M. Reisslein, "Trends in Optical Switching Techniques: A Short Survey," *IEEE Network*, vol. 22, no. 6, pp. 42–47, Nov. 2008.

[3] J. A. Hernández, V. López, J. L. G. Dorado, R. Nejabati, H. Overby, A. Rostami, K. Vlachos, and G. Zervas, "Optical Burst Switching," in *Enabling Optical Internet with Advanced Network Technologies*, J. Aracil and F. Callegati, Eds. London: Springer London, 2009, pp. 87–130.

[4] M. Imran, M. Collier, P. Landais, and K. Katrinis, "Software-Defined Optical Burst Switching for HPC and Cloud Computing Data Centers," *Journal of Optical Communications and Networking*, vol. 8, no. 8, p. 610, Aug. 2016.

[5] G. Franzl, M. F. Hayat, T. Holynski, and A. V. Manolova, "Burst-Switched Optical Networks Supporting Legacy and Future Service Types," *Journal of Computer Networks and Communications*, vol. 2011, pp. 1–17, 2011.

[6] A. Bhasin, M. Kaushik, and U. Sharma, "Wavelength Division Multiplexing Optical Networks," vol. 4, p. 4, 2016.

[7] D. H. Patel, "OPTICAL BURST SWITCHING NETWORK INTRODUCTION, CHALLENGES AND SOLUTION," vol. 1, no. 9, p. 9.

[8] N. Mubarakah, D. D. Fadhilah, and Suherman, "Point to Point Communication Link Design by Using Optical DWDM Network," in *2020 4rd International Conference on Electrical, Telecommunication and Computer Engineering (ELTICOM)*. Medan, Indonesia: IEEE, Sep. 2020, pp. 265–268.

[9] B. Mukherjee, "WDM Optical Communication Networks: Progress and Challenges," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 10, pp. 1810–1824, Oct. 2000.

[10] R. C. Alferness, H. Kogelnik, and T. H. Wood, "The Evolution of Optical Systems: Optics Everywhere," *Bell Labs Technical Journal*, vol. 5, no. 1, pp. 188–202, Aug. 2002.

[11] A. Tzanakaki, L. Zacharopoulos, and L. Tomkos, "Optical add/drop multiplexers and optical cross-connects for wavelength routed networks," in *Proceedings of 2003 5th International Conference on Transparent Optical Networks, 2003.*, vol. 1. Warsaw, Poland: IEEE, 2003, pp. 41–46.

[12] A. A. M. Saleh and J. M. Simmons, "All-Optical Networking—Evolution, Benefits, Challenges, and Future Vision," *Proceedings of the IEEE*, vol. 100, no. 5, pp. 1105–1117, May 2012.

[13] M. Guerreiro, N. S. Correia, and M. C. Medeiros, "Evaluation and Comparison of Signaling Reservation Protocols for Grid over OBS Networks Employing Active Routers," in *2007 9th International Conference on Transparent Optical Networks*. Rome, Italy: IEEE, Jul. 2007, pp. 105–108.

[14] M. Irshid and M. Kavehrad, "Distributed optical passive star couplers," *IEEE Photonics Technology Letters*, vol. 3, no. 3, pp. 247–249, Mar. 1991.

[15] M. Jonsson, "Optical Interconnection Technology in Switches, Routers and Optical Cross Connects," in *Proceedings International Conference on ParallelProcessing Workshops*. Valencia, Spain: IEEE Comput. Soc, 2001, pp. 319–326.

[16] J. S. Turner, "Terabit Burst Switching:," Defense Technical Information Center, Fort Belvoir, VA, Tech. Rep., Jan. 2003.

[17] L. Zhou, Z. Xu, X. Cheng, and Q. Huang, "An Optical Circuit Switching Network Architecture and Reconfiguration Schemes for Datacenter," *Optics Communications*, vol. 335, pp. 250–256, Jan. 2015.

[18] K. Barker, A. Benner, R. Hoare, A. Hoisie, A. Jones, D. Kerbyson, D. Li, R. Melhem, R. Rajamony, E. Schenfeld, S. Shao, C. Stunkel, and P. Walker, "On the Feasibility of Optical Circuit Switching for High Performance Computing Systems," in *ACM/IEEE SC 2005 Conference (SC'05)*. Seattle, WA, USA: IEEE, 2005, pp. 16–16.

[19] A. M. Balamurugan and D. A. Sivasubramanian, "OPTICAL BURST SWITCHING ISSUES AND ITS FEATURES," vol. 2, no. 3, p. 10, 2013.

[20] I. S. Niak, "Overview of Optical Packet Switching," p. 14.

[21] G. N. Rouskas and L. Xu, "Chapter 1 OPTICAL PACKET SWITCHING," 2004.

[22] Y. Xiong, M. Vandenhoute, H. C. Cankaya, and E. C. Road, "Design and analysis of optical burst-switched networks," p. 9.

[23] S. J. B. Yoo, "Optical Packet and Burst Switching Technologies for the Future Photonic Internet," *JOURNAL OF LIGHTWAVE TECHNOLOGY*, vol. 24, no. 12, p. 25, 2006.

[24] Y. Chen, C. Qiao, and X. Yu, "Optical Burst Switching (OBS): A New Area in Optical Networking Research," p. 13.

[25] T. Battestilli and H. Perros, "An introduction to optical burst switching," p. 6.

[26] M. L. Singh, K. Pubby, and D. Kapoor, "INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY," *International Journal of Engineering Sciences*, p. 9.

[27] C. Paar and J. Pelzl, *Understanding Cryptography: A Textbook for Students and Practitioners*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.

[28] J. A. Hernández, J. Aracil, V. López, and J. L. de Vergara, "On the analysis of burst-assembly delay in OBS networks and applications in delay-based service differentiation," *Photonic Network Communications*, vol. 14, no. 1, pp. 49–62, Aug. 2007.

[29] M. A. Dawood, M. Mahmoud, and M. H. Aly, "Adaptive data burst assembly in OBS networks," in *2016 Sixth International Conference on Digital Information Processing and Communications (ICDIPC)*. Beirut, Lebanon: IEEE, Apr. 2016, pp. 192–197.

[30] V. M. Vokkarane, Qiong Zhang, J. P. Jue, and Biao Chen, "Generalized Burst Assembly and Scheduling Techniques for QoS Support in Optical Burst-Switched Networks," in *Global Telecommunications Conference, 2002. GLOBECOM '02. IEEE*, vol. 3. Taipei, Taiwan: IEEE, 2002, pp. 2747–2751.

[31] B. Kantarci and S. Oktug, "Adaptive Threshold based Burst Assembly in OBS Networks," in *2006 Canadian Conference on Electrical and Computer Engineering*. Ottawa, Ont.: IEEE, May 2006, pp. 1419–1422.

[32] J. P. Jue and V. M. Vokkarane, *Optical Burst Switched Networks*, ser. Optical Networks Series. New York: Springer, 2005.

[33] R. Jankuniene and P. Tervydis, "The Contention Resolution in OBS Network," *Elektronika ir Elektrotechnika*, vol. 20, no. 6, pp. 144–149, Jun. 2014.

[34] A. K. Garg, "Contention reduction and service differentiation in OBS networks," *Optik*, vol. 123, no. 12, pp. 1108–1111, Jun. 2012.

[35] M. S. Reza, M. M. Hossain, and S. P. Majumder, "Contention Problem in Optical Burst Switching Network," in *2010 International Conference on Computational Intelligence and Communication Networks*. Bhopal: IEEE, Nov. 2010, pp. 239–242.

[36] A. K. Garg and R. Kaler, "An efficient scheme for optimizing channel utilization in OBS networks," *Optik*, vol. 121, no. 9, pp. 793–799, May 2010.

[37] A. Pantaleo, M. Tornatore, A. Pattavina, C. Raffaelli, and F. Callegati, "Considerations on In-Band and Out-of-Band Signalling Constraints in OBS Networks," in *2007 9th International Conference on Transparent Optical Networks*. Rome, Italy: IEEE, Jul. 2007, pp. 113–116.

[38] P. Kirci and A. H. Zaim, "Performance analysis of signaling protocols on OBS switches," in *Optics East 2005*, M. Atiquzzaman and S. I. Balandin, Eds., Boston, MA, Oct. 2005, p. 60110G.

[39] A. M. Kaheel, H. Alnuweiri, and F. Gebali, "A New Analytical Model for Computing Blocking Probability in Optical Burst Switching Networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 12, pp. 120–128, Dec. 2006.

[40] A. Gupta and H. Singh, "Comparison of Various Scheduling Techniques in OBS Networks," p. 7.

[41] S. Charcranoon, T. El-Bawab, H. Cankaya, and Jong-Dug Shin, "Group-scheduling for optical burst switched (OBS) networks," in *GLOBECOM '03. IEEE Global Telecommunications Conference (IEEE Cat. No.03CH37489)*, vol. 5. San Fransico, CA, USA: IEEE, 2003, pp. 2745–2749.

[42] M. Ljolje, R. Inkret, and B. Mikac, "A Comparative Analysis of Data Scheduling Algorithms in Optical Burst Switching Networks," in *Conference on Optical Network Design and Modeling, 2005*. Milan, Italy: IEEE, 2005, pp. 493–500.

[43] P. K. Chandra, A. K. Turuk, and B. Sahoo, "Survey on Optical Burst Switching in WDM Networks," in *2009 International Conference on Industrial and Information Systems ICIIS*. Peradeniya, Sri Lanka: IEEE, Dec. 2009, pp. 83–88.

[44] M. Sliti, M. Hamdi, and N. Boudriga, "A Novel Optical Firewall Architecture for Burst Switched Networks," *2010 12th International Conference on Transparent Optical Networks, ICTON 2010*, pp. 1–5, 2010.

[45] T. F. Fernandez, K. Brabagaran, and N. Sreenath, "Burstification Threat in Optical Burst Switched Networks," *International Conference on Communication and Signal Processing, ICCSP 2014 - Proceedings*, pp. 86–90, 2014.

[46] N. Sreenath, K. Muthuraj, and K. Brabagaran, "Burst Circulating Threat in Optical Burst Switched Networks," in *2014 IEEE International Conference on Advanced Communications Control and Computing Technologies*. Ramanathapuram, India: IEEE, May 2014, pp. 780–783.

[47] A. Detti, V. Eramo, and M. Listanti, "Performance evaluation of a new technique for IP support in a WDM optical network: Optical composite burst switching (OCBS)," *Journal of Lightwave Technology*, vol. 20, no. 2, pp. 154–165, Feb./2002.

[48] A. Abdelouahab and A. F. Mohammed, "On The Control Packet Structure of Optical Burst Switching Networks," p. 6.

[49] S. Mehraban, K. B. Vora, and D. Upadhyay, "Deploy Multi Protocol Label Switching (MPLS) Using Virtual Routing and Forwarding (VRF)," in *2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI)*. Tirunelveli: IEEE, May 2018, pp. 543–548.

[50] M. Jin and O. W. W. Yang, "On Controlling the Loss Rate in Optical Burst Switching Networks," in *2006 3rd International Conference on Broadband Communications, Networks and Systems*. San Jose, CA, USA: IEEE, Oct. 2006, pp. 1–9.

[51] N. Sreenath, K. Muthuraj, and P. Sivasubramanian, "Secure Optical Internet: Attack Detection and Prevention Mechanism," in *2012 International Conference on Computing, Electronics and Electrical Technologies ICCEET)*. Nagercoil, Tamil Nadu, India: IEEE, Mar. 2012, pp. 1009–1012.

[52] V. Shakhov, "DDoS Flooding Attacks in OBS Networks," *Proceedings - 2012 7th International Forum on Strategic Technology, IFOST 2012*, pp. 1–4, 2012.

[53] S. S. Chouhan and S. Sharma, "Identification of current attacks and their counter measures in Optical Burst Switched (OBS) network," *International Journal of Advanced Computer Research*, vol. 2, no. 1, p. 5, Mar. 2012.

[54] Y. Chen, P. K. Verma, and S. Kak, "Embedded Security Framework for Integrated Classical and Quantum Cryptography Services in Optical Burst Switching Networks," *Security and Communication Networks*, pp. n/a–n/a, 2009.

[55] P. S. Subramanian and K. Muthuraj, "Threats in Optical Burst Switched Network," vol. 2, p. 5.

[56] N. Sreenath, K. Muthuraj, and K. Brabagaran, "Burst Circulating Threat in Optical Burst Switched Networks," *Proceedings of 2014 IEEE International Conference on Advanced Communication, Control and Computing Technologies, ICACCCT 2014*, no. 978, pp. 780–783, 2015.

[57] P. Merlin Sweetha Vinola, A. M. Balamurugan, V. Chandana, and V. Durairaji, "Enhanced HMAC Structure Based Burst Header Authentication Design for Optical Burst Switched Networks," *RTEICT 2017 - 2nd IEEE International Conference on Recent Trends in Electronics, Information and Communication Technology, Proceedings*, vol. 2018-Janua, pp. 409–411, 2017.

[58] Y. Chen and P. K. Verma, "Secure Optical Burst Switching: Framework and Research Directions," *IEEE Communications Magazine*, vol. 46, no. 8, pp. 40–45, Aug. 2008.

[59] Y. Coulibaly, A. A. I. Al-Kilany, M. S. A. Latiff, G. Rouskas, S. Mandala, and M. A. Razzaque, "Secure Burst Control Packet Scheme for Optical Burst Switching Networks," *IBP 2015 - 2015 IEEE International Broadband and Photonics Conference*, no. April, pp. 86–91, 2015.

[60] M. Zahid Hasan, K. M. Zubair Hasan, and A. Sattar, "Burst Header Packet Flood Detection in Optical Burst Switching Network Using Deep Learning Model," *Procedia Computer Science*, vol. 143, pp. 970–977, 2018.

[61] H. Kaur and S. Singh, "Prevention of DDOS in Optical Burst Switching Using Genetic Algorithm," *Indian Journal of Science and Technology*, vol. 9, no. 36, Sep. 2016.

[62] R. Alshboul, "Flood Attacks Control in Optical Burst Networks by Inducing Rules Using Data Mining," p. 8, 2018.

[63] A. Rajab, C.-T. Huang, and M. Al-Shargabi, "Decision tree rule learning approach to counter burst header packet flooding attack in Optical Burst Switching network," *Optical Switching and Networking*, vol. 29, pp. 15–26, Jul. 2018.

[64] P. M. S. Vinola, A. M. Balamurugan, V. Chandana, and V. Durairaji, "Enhanced HMAC Structure Based Burst Header Authentication Design for Optical Burst Switched Networks," in *2017 2nd IEEE International Conference on Recent Trends in Electronics Information & CommunicationTechnology RTEICT*. Bangalore: IEEE, May 2017, pp. 409–411.

[65] V. N. G. J. Soares, I. D. C. Veiga, and J. J. P. C. Rodrigues, "OBS Simulation Tools: A Comparative Study," in *ICC Workshops - 2008 IEEE International Conference on Communications Workshops*. Beijing, China: IEEE, May 2008, pp. 256–260.

[66] G. Guirel, O. Alparslan, and E. Karasan, "NOBS: An ns2 based simulation tool for performance evaluation of TCP traffic in OBS networks," *Annals of Telecommunications*, vol. 62, pp. 618–632, May 2007.

[67] D. Rajani, "Importance of Cryptography in Network Security," p. 6, Apr. 2013.

[68] A. M. Qadir and N. Varol, "A Review Paper on Cryptography," in *2019 7th International Symposium on Digital Forensics and Security (ISDFS)*. Barcelos, Portugal: IEEE, Jun. 2019, pp. 1–6.

[69] A. Karakra and A. Alsadeh, "A-RSA: Augmented RSA," in *2016 SAI Computing Conference (SAI)*. London, United Kingdom: IEEE, Jul. 2016, pp. 1016–1023.

[70] L. K. Galla, V. S. Koganti, and N. Nuthalapati, "Implementation of RSA," in *2016 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*. Kumaracoil, India: IEEE, Dec. 2016, pp. 81–87.

[71] Yunfei Li, Qing Liu, and Tong Li, "Design and implementation of an improved RSA algorithm," in *2010 International Conference on E-Health Networking Digital Ecosystems and Technologies (EDT)*. Shenzhen, China: IEEE, Apr. 2010, pp. 390–393.

[72] I. Andrea, C. Chrysostomou, and G. Hadjichristofi, "Internet of Things: Security vulnerabilities and challenges," in *2015 IEEE Symposium on Computers and Communication (ISCC)*. Larnaca: IEEE, Jul. 2015, pp. 180–187.

[73] F. Xu, M. Curty, B. Qi, and H.-K. Lo, "Measurement-Device-Independent Quantum Cryptography," *IEEE JOURNAL OF SELECTED TOPICS IN QUANTUM ELECTRONICS*, vol. 21, no. 3, p. 11, 2015.

[74] D. H. Van and N. D. Thuc, "A Privacy Preserving Message Authentication Code," in *2015 5th International Conference on IT Convergence and Security (ICITCS)*. Kuala Lumpur, Malaysia: IEEE, Aug. 2015, pp. 1–4.

[75] B. Preneel and P. van Oorschot, "On the security of iterated message authentication codes," *IEEE Transactions on Information Theory*, vol. 45, no. 1, pp. 188–199, Jan./1999.

[76] D. Chen, S. Jiang, and Z. Qin, "Message Authentication Code over a wiretap channel," in *2015 IEEE International Symposium on Information Theory (ISIT)*. Hong Kong, Hong Kong: IEEE, Jun. 2015, pp. 2301–2305.

[77] R. Haddaji, "Comparison of Digital Signature Algorithm and Authentication Schemes for H . 264 Compressed Video," vol. 7, no. 9, pp. 357–363, 2016.

[78] R. Sobti and G. Ganesan, "Cryptographic Hash Functions: A Review," *International Journal of Computer Science Issues, ISSN (Online): 1694-0814*, vol. Vol 9, pp. 461–479, Mar. 2012.

[79] S. Debnath, A. Chattopadhyay, and S. Dutta, "Brief review on journey of secured hash algorithms," in *2017 4th International Conference on Opto-Electronics and Applied Optics (Optronix)*. Kolkata: IEEE, Nov. 2017, pp. 1–5.

[80] I. B. Damgard, "A Design Principle for Hash Functions," in *Advances in Cryptology — CRYPTO' 89 Proceedings*, G. Brassard, Ed. New York, NY: Springer New York, 1990, vol. 435, pp. 416–427.

[81] S. Mathew and K. P. Jacob, "Performance Evaluation of Popular Hash Functions," vol. 4, no. 1, p. 4, 2010.

[82] A. A. Alkandari, I. F. Al-Shaikhli, and M. A. Alahmad, "Cryptographic Hash Function: A High Level View," in *2013 International Conference on Informatics and Creative Multimedia*. Kuala Lumpur, Malaysia: IEEE, Sep. 2013, pp. 128–134.

[83] R. C. Merkle, "One Way Hash Functions and DES," in *Advances in Cryptology — CRYPTO' 89 Proceedings*, G. Goos, J. Hartmanis, D. Barstow, W. Brauer, P. Brinch Hansen, D. Gries, D. Luckham, C. Moler, A. Pnueli, G. Seegmüller, J. Stoer, N. Wirth, and G. Brassard, Eds. New York, NY: Springer New York, 1990, vol. 435, pp. 428–446.

[84] *The MD4 Message-Digest Algorithm*, Apr. 1992.

[85] N. I. o. S. and Technology, "Secure Hash Standard (SHS)," U.S. Department of Commerce, Tech. Rep. Federal Information Processing Standard (FIPS) 180 (Withdrawn), May 1993.

[86] Rivest, *RFC1321: The MD5 Message-Digest Algorithm | Guide Books*.

[87] N. I. o. S. and Technology, "Secure Hash Standard," U.S. Department of Commerce, Tech. Rep. Federal Information Processing Standard (FIPS) 180-1 (Withdrawn), Apr. 1995.

[88] ——, "Secure Hash Standard (SHS)," U.S. Department of Commerce, Tech. Rep. Federal Information Processing Standard (FIPS) 180-2 (Withdrawn), Feb. 2004.

[89] H. Dobbertin, A. Bosselaers, and B. Preneel, "RIPEMD-160: A Strengthened Version of RIPEMD," Feb. 2000.

[90] N. P. Smart, *Cryptography Made Simple*, ser. Information Security and Cryptography. Cham: Springer International Publishing, 2016.

[91] H. Tiwari, "Merkle-Damgard Construction Method and Alternatives: A Review," vol. 41, no. 2, p. 22, 2017.

[92] W. Stallings, *Cryptography and Network Security (4th Edition)*.

[93] M. Knezzevic, K. Sakiyama, Y. K. Lee, and I. Verbauwhede, "On the high-throughput implementation of RIPEMD-160 hash algorithm," in *2008 International Conference on Application-Specific Systems, Architectures and Processors*. Leuven, Belgium: IEEE, Jul. 2008, pp. 85–90.

[94] Chiu-Wah Ng, Tung-Sang Ng, and Kun-Wah Yip, "A unified architecture of MD5 and RIPEMD-160 hash algorithms," in *2004 IEEE International Symposium on Circuits and Systems (IEEE Cat. No.04CH37512)*. Vancouver, BC, Canada: IEEE, 2004, pp. II–889–92.

[95] A. D. Threats, "A Study of Data Threats and the Role of Cryptography Algorithms," *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pp. 819–824, 2018.

[96] M. Faheem, S. Jamel, A. Hassan, Z. A., N. Shafinaz, and M. Mat, "A Survey on the Cryptographic Encryption Algorithms," *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 11, 2017.

[97] M. Panda, "Performance analysis of encryption algorithms for security," in *2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES)*. Paralakhemundi, Odisha, India: IEEE, Oct. 2016, pp. 278–284.

[98] S. Chandra, S. Paira, S. S. Alam, and G. Sanyal, "A comparative survey of Symmetric and Asymmetric Key Cryptography," in *2014 International Conference on Electronics, Communication and Computational Engineering (ICECCE)*. Hosur, Tamilnadu, India: IEEE, Nov. 2014, pp. 83–93.

[99] M. Smid and D. Branstad, "Data Encryption Standard: Past and future," *Proceedings of the IEEE*, vol. 76, no. 5, pp. 550–559, May 1988.

[100] L. Ali, N. Yunus, H. Jaafar, R. Wagiran, and E. Low, "Implementation of triple data encryption algorithm using VHDL," in *2004 IEEE International Conference on Semiconductor Electronics*. Kuala Lumpur, Malaysia: IEEE, 2004, p. 5 pp.

[101] F. J. D'souza and D. Panchal, "Advanced encryption standard (AES) security enhancement using hybrid approach," in *2017 International Conference on Computing, Communication and Automation (ICCCA)*. Greater Noida: IEEE, May 2017, pp. 647–652.

[102] M. N. Valmik and P. V. K. Kshirsagar, "Blowfish Algorithm," *IOSR Journal of Computer Engineering*, vol. 16, no. 2, pp. 80–83, 2014.

[103] E. Hyytiä, "Resource Allocation and Performance Analysis Problems in Optical Networks," PhD Thesis, Helsinki University of Technology, Espoo, 2004.

[104] X. Zhang and K. Parhi, "High-Speed VLSI Architectures for the AES Algorithm," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 9, pp. 957–967, Sep. 2004.

[105] Ritambhara, A. Gupta, and M. Jaiswal, "An Enhanced AES Algorithm Using Cascading Method on 400 Bits Key Size Used in Enhancing the Safety of next Generation Internet of Things (IOT)," in *2017 International Conference on Computing, Communication and Automation (ICCCA)*, May 2017, pp. 422–427.

[106] J. D. Dalal, S. S. Dayala, and N. Shah, "Optimized AES Algorithm Using Galois Field Multiplication and Parallel Key Scheduling," in *2012 1st International Conference on Emerging Technology Trends in Electronics, Communication Networking*, Dec. 2012, pp. 1–5.

[107] J. Vashishtha, "Evaluating the Performance of Symmetric Key Algorithms : AES (Advanced Encryption Standard) and DES (Data Encryption Standard)," vol. 15, no. 4, pp. 43–49, 2012.

[108] M. Ebrahim, "Symmetric Algorithm Survey : A Comparative Analysis," vol. 61, no. 20, pp. 12–19, 2013.

[109] O. Dunkelman, G. Sekar, and B. Preneel, "Improved Meet-in-the-Middle Attacks on Reduced-Round DES," in *Progress in Cryptology – INDOCRYPT 2007*, K. Srinathan, C. P. Rangan, and M. Yung, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, vol. 4859, pp. 86–100.

[110] "Comparison of ciphers," http://www.javamex.com/tutorials/cryptography/ciphers.shtml.

[111] O. Kara and C. Manap, "A New Class of Weak Keys for Blowfish," in *Fast Software Encryption*, A. Biryukov, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, vol. 4593, pp. 167–180.

[112] E. Rescorla, "Diffie-Hellman Key Agreement Method," RFC Editor, Tech. Rep. RFC2631, Jun. 1999.

[113] N. M. Noor, N. M. Din, E. Ahmed, and A. N. A. Kadir, "OMNET++ based cognitive radio simulation network," in *2016 7th IEEE Control and System Graduate Research Colloquium (ICSGRC)*. Shah Alam: IEEE, Aug. 2016, pp. 28–33.

[114] F. Loukdache, S. Elhajji, K. Salah-ddine, and A. el Ghazi, "Modeling an OBS Network in OMNeT + + and the Impact of Data Channels in Such Network," *Lecture Notes in Engineering and Computer Science*, vol. 2198, Jul. 2012.