SECURITY IN ROBOT OPERATING SYSTEM (ROS) BY USING ADVANCED ENCRYPTION STANDARD (AES)

NOR ALIA SYUHADA BINTI SHAHARUDIN

Bachelor of Computer Science (Computer Systems & Networking)

UNIVERSITI MALAYSIA PAHANG

UNIVERSITI MALAYSIA PAHANG

DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name : NOR ALIA SYUHADA BINTI SHAHARUDIN

Date of Birth : 04 DECEMBER 1996

Title : SECURITY IN ROBOT OPERATING SYSTEM (ROS) BY USING ADVANCED ENCRYPTION STANDARD (AES)

Academic Session : SEM I 2018/2019

I declare that this thesis is classified as:

☐	CONFIDENTIAL	(Contains confidential information under the Official Secret Act 1997)*
☐	RESTRICTED	(Contains restricted information as specified by the organization where research was done)*
☑	OPEN ACCESS	I agree that my thesis to be published as online open access (Full Text)

I acknowledge that Universiti Malaysia Pahang reserves the following rights:

1. The Thesis is the Property of Universiti Malaysia Pahang
2. The Library of Universiti Malaysia Pahang has the right to make copies of the thesis for the purpose of research only.
3. The Library has the right to make copies of the thesis for academic exchange.

Certified by:

(Student's Signature)

961204-145192
New IC/Passport Number
Date: 24 DECEMBER 2018

(Supervisor's Signature)

DR. LUHUR BAYUAJI
Name of Supervisor
Date: 24 DECEMBER 2018

NOTE : * If the thesis is CONFIDENTIAL or RESTRICTED, please attach a thesis declaration letter.

SUPERVISOR'S DECLARATION

I hereby declare that I have checked this thesis and in my opinion, this thesis is adequate in terms of scope and quality for the award of the degree in Bachelor of Computer Science (Computer Systems & Networking)

(Supervisor's Signature)

Full Name : DR LUHUR BAYUAJI

Position : SENIOR LECTURER

Date : 24 DECEMBER 2018

STUDENT'S DECLARATION

I hereby declare that the work in this thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Universiti Malaysia Pahang or any other institutions.

_____

(Student's Signature)

Full Name : NOR ALIA SYUHADA BINTI SHAHARUDIN

ID Number : CA15073

Date : 24 DECEMBER 2018

SECURITY IN ROBOT OPERATING SYSTEM (ROS) BY USING ADVANCED ENCRYPTION STANDARD (AES)

NOR ALIA SYUHADA BINTI SHAHARUDIN

Thesis submitted in fulfillment of the requirements
for the award of the degree of
Bachelor of Computer Science (Computer Systems & Networking)

Faculty of Computer Systems & Sofware Engineering

UNIVERSITI MALAYSIA PAHANG

DECEMBER 2018

ACKNOWLEDGEMENTS

I would like to express my appreciation to my supervisor, Dr. Luhur Bayuaji for his guidance during this research is conducted. Without his valuable assistance, this thesis would not have been completed.

In addition, I also would like to thank to all the people around me who involve directly or indirect in completing this research. I also appreciate all the moral support and courage that has been given to me from my family members and friends who always be there to help me finishing this research as well as to those who sacrifice their time to help me improve and complete this research.

Finally, I would like to thanks to the staff of the Faculty of Computer Systems & Software Engineering for their valuable assistance during the development of this thesis.

ABSTRAK

Oleh kerana system robotik semakin tersebar, siber sekuriti muncul sebagai tumpuan utama. Pada masa kini, kebanyakan sistem autonomi dibina menggunakan rangka ROS yang merupakan rangka yang paling popular untuk membangunkan aplikasi robotik bersama dengan perisian komersil yang lain. ROS adalah rangka kerja yang diedarkan di mana nod menerbitkan maklumat yang digunakan oleh nod lain. Model ini memudahkan komunikasi data tetapi menimbulkan ancaman utama kerana proses yang berniat jahat boleh mengganggu komunikasi dengan mudah, membaca mesej peribadi, atau mengambil alih pergerakan robot. Kajian ini menyiasat prestasi robot apabila memecahkan mesej yang ditukar antara nod ROS di bawah paradigma menerbitkan/melanggan. Khususnya, kajian ini memberi tumpuan kepada penggunaan algoritma penyulitan iaitu AES. Algoritma penyulitan akan dinilai berdasarkan prestasi sistem, baik dari sudut pengkomputeran dan juga komunikasi.

ABSTRACT

As robotic systems spread, cybersecurity emerges as major concern. Currently, most research autonomous systems are built using the ROS framework which has become the most popular framework for developing robotic applications along with other commercial software. ROS is a distributed framework where nodes publish information that other nodes consume. This model simplifies data communication but poses a major threat because a malicious process could easily interfere the communications, read private messages, or even take over the robots movement. The study investigates a robot's performance when ciphering the messages interchanged between ROS nodes under the publish/subscribe paradigm. In particular, this research focuses on using encryption algorithm which is AES. The encryption algorithm will be evaluated according to the performance of the system, both from computing and communications point of view.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

3DES	Triple DES
AES	Advanced Encryption Standard
CBC	Cipher Block Chaining
CFB	Cipher Feedback
CTR	Counter
DoS	Denial of Service
ECB	Electronic Code Book
OFB	Output Feedback
ROS	Robot Operating System

CHAPTER 1

INTRODUCTION

1.1 Background of Study

In current world growing technology, robots become more sophisticated and it have a high demand in fields such as medical, construction, military as well as household robots. Robots are playing an important role for human that can affect our daily lives. National security and defense are now depend on drones. Amazon use robots to distribute their supply chain and goods. Surgical robots that is operated remotely by the surgeon will likely to be used in more scenarios such as emergency response and military. Automated vehicles are being developed by companies like Google and Tesla. An analysis of 280 companies that was carried out by Department of Commerce on Competitiveness showed an average rate growth of 62% in the healthcare and eldercare markets as well as 20% average growth rate for robotics use in manufacturing, service and medical fields (Dorsey, Martin, Howard, & Coovert, 2017). Based on this data, it proves that the use of the robots is increasing and will be likely to contribute more in the future.

Robotics systems are growing not just in the virtual world, science-fiction movies, but also in our normal life. It is possible to find driverless cars in the streets, autonomous robotic guides at museums and the most common is autonomous vacuum cleaners in our homes. These robotic systems can suffer from different types of cyber-attack, hence some standard of cybersecurity is enforced.

The widely known framework for developing robotic applications is ROS (Robotic Operating System) that controls the autonomous behaviour of the robots. ROS is an emerging standard for creating a new robots application in the future, but it is not

immune to cyber-attack or hacking. Therefore, it is important for the ROS framework to be more secured to avoid security problem.

1.2 Problem Statement

Generally, robots are created to provide service to people, hence the human robot interaction is important. It can obtain user's information in a blink of eye with the advancement of technology in the last few decades. Service robots may one day also collect data and information about the health and wellbeing of the person that it serves. The data that non-authorized and suspicious entitiy gained will let them to misuse it for their own benefit. The robots also can be controlled if outsider able to enter the system of the robots.

ROS is a distributed framework where nodes publish information that other nodes consume. The message-passing distribution between the nodes in ROS was implemented in plain text. It simplifies data communication but poses major threat because malicious process could easily interfere the communications and read private messages. Therefore, cryptography technique such as Advanced Encryption Standard (AES) must be implemented in publisher and subscriber nodes to protect the confidentiality, integrity and availability of the data.

In addition, cryptography is an important technology that able to protect information against the outsider such as suspicious users and adversaries. The fundamental issue in plotting and designing an encryption algorithm must be the security of the algorithm against undesirable attack. AES have three different length of keys which is 128, 192 and 256 bits. The differences in key length will present different performance. Hence, AES with different key length is analysed in order to choose which algorithm is better for ROS framework.

1.3 Objective

Based on the problems statement, the objectives of this research are:

i. To identify the possible cyber security attack that can occur in Robot Operating System (ROS).
ii. To implement the AES algorithm with different key length in the communication of ROS.
iii. To evaluate the performance of AES algorithm based on the system parameter in ROS.

1.4 Scope

The study focus on comparison between different length of keys in AES algorithm in the security of robots based on ROS. The purpose of the algorithm is to improve the security of the communication in ROS. The scope also includes the user such as researchers and organizations. The main software that is used in this research is ROS Indigo.

1.5 Thesis Organization

The thesis consists of five chapters. The organization and flow of the thesis is as follows. Chapter 1 shall discuss on introduction to the research. In the second chapter, literature review of the research is discussed. In Chapter 3, the methodology used is interpreted. Chapter 4 are literally about the implementation and result discussion. Lastly, Chapter 5 contains the conclusion of the research findings.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

In Chapter 1, the research introduction have been discussed which consist of the problem statement, objective and scope. In this chapter, the relevant literature review will be discussed to understand about Robot Operating System (ROS), cryptography method and how it works. Hence, comparison between the method will be further elaborated to justify the current work.

2.2 Background on Robots

A robot is a machine that is programmable by a computer that capable of carrying out a complex series of actions automatically. Robots can be defined as a combination of mechanical structures, sensors, actuators, and computer software that manages and controls these devices (Morante, Victores, & Balaguer, 2015). Robots might be established to take on a human form but most robots are machines that are intended to perform a task with no regard on how they look.

Robots are already showing up in thousands of homes and business. All signs as toys for children, companions for the elderly, customer assistants at stores and healthcare attendants. Robots will fill varying roles of service, as home and business assistants, physical companions, manufacturing workers, security and law enforcement and more.

As many of these smart machines are self-propelled and able to move without the help of human. It is important that they are secure, well protected, and not easy to hack. If not, instead of being a helpful resources to human, they could quickly become

dangerous tools that are capable to cause havoc and causing substantive harm to their surroundings that they are designed to serve.

2.3 Robot Operating System (ROS)

(Quigley et al., 2009) The Robot Operating System (ROS) is a popular distributed framework for creating robotic applications. It began as a research project for framework, but currently most of the manufacturers use ROS for building robotic applications. It is mostly used in industrial applications such as automotive, healthcare and manufacturing. For instance, object-manipulation robots like Baxter (by Rethink robotics) and service robots which is RB1 (by Robotnik) are both using ROS as a platform. ROS provides a communication layer remotely above a host operating system to construct a heterogeneous complete cluster for robots. It was developed to interpret and simply the code reuse among the robots that have wildly varying hardware and to support large-scale software integration efforts as the systems grow even more complex.

ROS are similar to any other operating system services that are equipped with a set of libraries for robotics which consists of hardware abstraction for sensors and actuators, low-level device control and inter-process communication that are functioning to control the robots ("ROS_Introduction - ROS Wiki," n.d.). Nodes, messages, topics and services are the essential and basic concepts of the ROS implementation. Computation takes place in ROS processes named nodes. ROS framework is basically a message-passing distributed system in which the nodes can send and receive messages. The architecture of this framework is based on publishing messages to topics processes. (Rodr, Casado, Fern, & Mart, 2016) For example, a process (node) can governing sensor accessed, perform the processing and publishing information as an information structure on the topic. Other process can subscribe to this topic to read the information. The process can decide and make a selection on the activity and the movement of the robot. Then, the nodes broadcast the commands in another topic to send them to the motors.

Common ROS configuration is consist by at least one ROS Master and some clients. ROS Master is the important root in the ROS system. Registration information about all of the topics and services used by ROS nodes can be managed by ROS Master. To register its information, nodes communicate with the Master and it gets the information of other registered nodes to ensure that it can authorize new connections with

their topics accordingly. Inbound connections are received via a network communication protocol; TCP Server Socket along with a header containing information about the messages such as data type, routing information, the name of the sending data node, the name of the Topic where the subscriber is connecting to and others. Figure 2.1 presents the model illustrated by ROS Technical Overview.

ROS binary packages is available for public to be downloaded. However, since it is developer-friendly software ecosystem, it is possible for the system to be the aim for cyber attacks later on due to the expanding level of connectivity and availability to outside services and site. The transmission of messages between the publishers and subscribers was implemented in plain text, which make it easy for the third party to access and gain data or modify the robot behavior just by connecting the same network. Therefore, the major security problems can easily be identified in ROS. The vulnerabilities includes plain-text communication, unprotected TCP ports, unencrypted data storage and XML-RPC legacy issues.

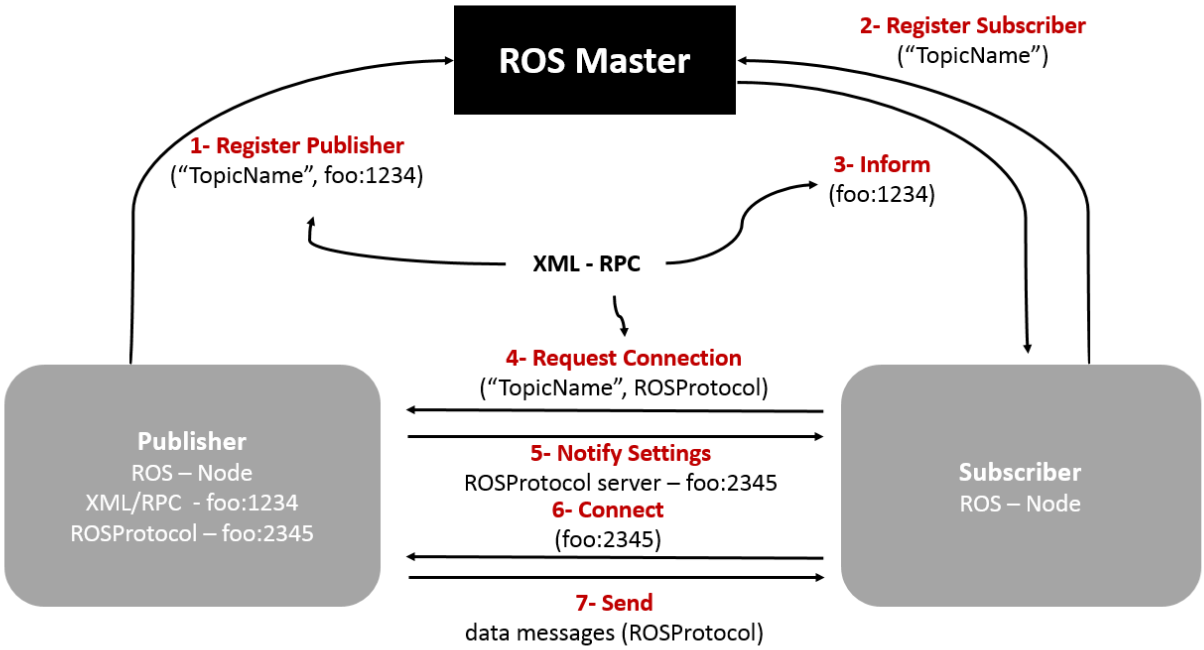

Figure 2.1 Conceptual model of ROS

2.4 Security Issues in Robot Operating System (ROS)

(Rodr et al., 2016) There are three basic types of security issues that may threaten the robotics system which are integrity (data modification), availability (data interruption) and confidentiality (data interception). These concept can make the robot to perform undesired behaviours or offer all information available in the scenario to the attacker. An example of integerity attack is the attacker manipulate the data and traffic between the user and the robot. User is teleoperating a robot and wants the robot to turn left, but an attacker intercept the command and forces the robot to move to the right instead. In this case, there would be an issue where the robots could cause a physical damage including falls or collission.

Robot Operating System (ROS) is one of the most popular open source frameworks and libraries that is used in industrial application. Since ROS is an open source, it may suffer from many known cybersecurity problems, such as clear text communication, authentication issues, and weak authorization schemes. All of these issues make robots insecure. Below are the some possible attack vectors that can occur in ROS:

2.4.1 Unauthorized Publishing (Injections)

A node in ROS may broadcast and distribute data for a random topic without the approval from the owner of the system. This issue can caused the attacker to inject the data or commands into the application purposely in order to interrupt the operation. For instance, fake movement commands might be injected into the robotic system leading to random motion that may hurt nearby persons or damage equipment. Another example is false sensor data might be injected to fake a normal system after a manipulation of the system or to take control the activity of the robot.

2.4.2 Unauthorized Data Access

Each node in ROS can subscribe to every topic in the system implementation. Then, it will accept any data that is published for the certain topic. This data can consist of private information such as business-critical information or details about the

communication process of the robots. This kind of attack is difficult to identify and detect since the node itself has no outgoing ROS communication.

2.4.3 Denial of Service (DoS) attack

Denial of Service is a cyber attack in which the attacker attempt to make the robots or network resource unavailable for the users by breaching the services of a host connected to the Internet. In this case, huge amount of fake data can easily be executed in ROS. Therefore, it leads to a high processing load on all nodes and will likely make the robots unable to perform meaningful process. Every node in the network is used to publish data for a topic in a subscribed targeted node since there is no control over which node may publish what kind of data. Hence, the node can be used for the DoS to launch its attack.

2.5 Cryptography

Cryptography is the process of converting or translating data to a format that is unreadable without the help of a tool or additional information. Cryptography is the science of designing the methods that allow data to be sent in a protected form and it is an important technology that can protect private information from third parties such as malicious users and attacker. The immensely use of networking leads to the data exchange over the network while communicating to different system. It is important to encrypt the message to ensure that intruder is unable to view and interpret the message. Various type of encryption algorithms are now widely used in information security to secure the data and information in the system (Hwang & Liu, 2005). In fundamental, cryptography is a way of hiding information by encrypting the message. The conversion of protecting information (encryption) into an unreadable format (encrypted text) known as cipher text. Only those who knows a secret key can decrypt the message into plain text. Figure 2.2 shows how encryption process works.

Figure 2.2 Encryption

As the Internet and different types of electronic communication become more established and powerful, the security is becoming increasingly important. The major function of any cryptographic algorithm is to establish maximal stability of the crypto-ciphered information includes robotic middleware which is Robot Operating System (ROS). In this research, cryptographic algorithms that will be discussed are 3DES (Triple Data Encryption Standard), AES (Advanced Encryption Standard) and Blowfish.

2.5.1 Basic Terminology Used in Cryptography

(Bhanot & Hans, 2015) There are a few terms that we need to understand in the encryption process due to in every algorithm description, these common terms are going to be discussed:

- **Cipher Text**

The plain text is encrypted in unreadable message. This unreadable message is called as Cipher Text. Example: “Hello” message is converted into “@#$*^”.

- **Encryption**

Encryption is the process converting Plain text into Cipher text. This message cannot be read, hence it can secure the communication over the network. Encryption algorithm is applied into the process.

- **Decryption**

This process is the contradiction or opposite process of the encryption. Specific algorithm is used to convert the cipher text into plain text.

- **Key**

It can be categorized into two type of keys which are Symmetric (private) and Asymmetric (public) keys. In Symmetric keys encryption, one key is used to encrypt and decrypt the message. Meanwhile in Asymmetric keys, two keys are used which includes private and public keys.

In a public-key encryption scheme, the public key is developed by some servers as an encryption key. Any individual that have the public key can use the key to encrypt the messages and form the cipher-text. Private key act as a decryption key and it is used by anyone who knows it to retrieve the original message from any cipher-text generated using the matching public key. The stealth of encrypted message is secured against attacker who knows and have the encryption key.

- **Key Size**

Key size is the measurement of length of key in bits, used in any algorithm.

- **Block Size**

Key cipher works on fixed length string of bits. This fix length of string in bits is called Block size. The block size depends on the algorithm.

- **Plain Text or Normal Text**

The original text or message used in the communication is known as plain text. Example: Alex sends “Hello” to John. “Hello” is the plain text.

- **Round**

Round of encryption means that how long the time is needed for the encryption algorithm to execute in complete encryption process until it produce cipher text as output.

2.6 Overview of Various Algorithms

2.6.1 3DES (Triple Data Encryption Standard)

3DES (Triple DES) is an encryption algorithm that is similar to DES. It is replacement for DES because of the advancement in key searching. It was standardized in ANSI X9.17 & ISO 8732 and in PEM for key management. The encryption method is identical with DES but it applied three times to every data block to increase the encryption level and the average safe time. It uses the simple DES encryption algorithm thrice to improve the security of encrypted text.

Based on Figure 2.3, same data is encrypted twice using DES. Therefore, it makes the encryption more secure and hard to break. This encryption algorithm is essentially a Block Cipher that use 48 rounds (three times DES) in its computation, and the key length is 168 bits. Each block contains 64 bits of data.

3DES is supposed to be protected up to at least 2^{112} security, however, in terms of performance, some research stated that 3DES is slower than other methods, especially in software computations (Stallings, n.d.). 3DES also supply sufficient security. That is the reason users needed the successor of 3DES.

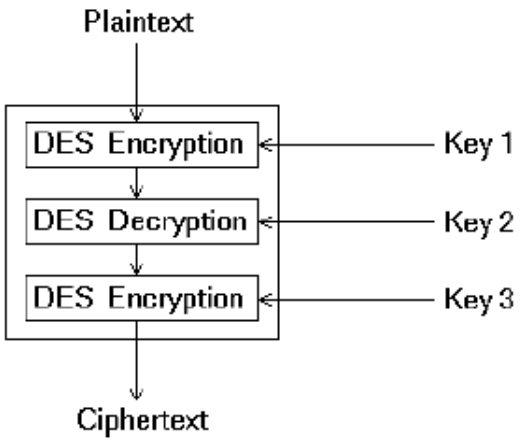

Figure 2.3 3DES Structure

The advantages of 3DES is that it is three times secure because as stated before, it is combination of three DES encryption algorithms with various keys at each level. This algorithm is chosen over simple DES encryption algorithm. Even though it brings the sufficient security to the data, but it is not perfect because it use a lot of time as well as the speed is also slower than DES encryption algorithm.

There are several modes:

- **DES-EDE3**

Encrypt, Decrypt and Encrypt with three unique keys as mentioned in Figure 2 (Key 1, Key 2, Key 3).

- **DES-EEE3**

A block of data in encrypted twice with a different key and encrypted again with another key, with using three unique keys.

- **DES-EDE2 and DES-EEE2**

Use two keys only, where first and last encryption is done using the same key.

2.6.2 AES (Advanced Encryption Standard)

In 1997, the National Institute of Standards and Technology (NIST) announced to choose a replacement to DES. Advanced Encryption Standard (AES) that is developed by Vincent Rijmen, Joan Daeman in 2001 was selected to be the successor to DES. AES is a symmetric block cipher that contains variable key length of 128, 192 or 256 bits. Each cipher is capable to encrypt and decrypt data blocks of 128 bits in 10,12 and 14 rounds depends on the size of the key. There are 10 rounds for 128-bit keys, 12 rounds for 192-bit keys, and 14-rounds for 256-bit keys (Shirabadagi & Nadagoud, 2017). Figure 2.4 shows the AES algorithm.

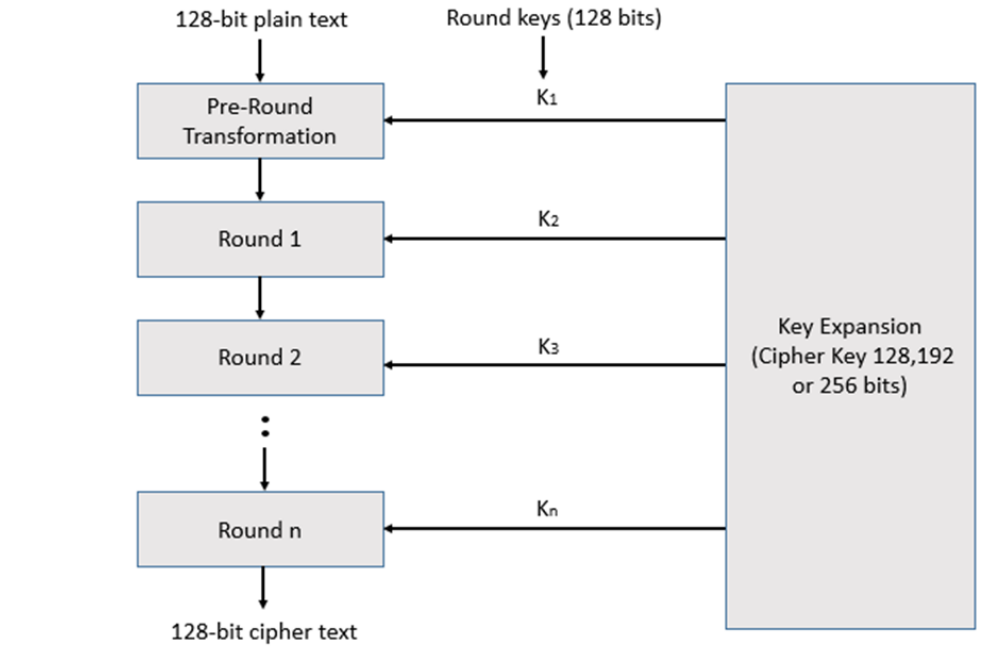

Figure 2.4 AES algorithm

Except for the last round, all of the other rounds are the same. Each round in the encryption process follow some steps to complete the process until it reach (n). Each round have four steps which are Substitute Bytes, Shift Rows, Mix Column and Add Round Key as shown in Figure 2.5. Table 2.1 explain the details of the four steps in AES algorithm.

Figure 2.5 AES Round Steps

Table 2.1 AES Steps

Steps	Description
Substitution Round	Sub-Bytes are byte-by-byte substitution during the encryption process.
Shift Rows	Shift the rows of the state array during the forward process (S-Box process).
Mix Column	During the forward process, mix up the bytes in each column separately.
Add Round Key	Round key is added to the output of the previous step during the forward process. This step is quite different from others because of the difference in key size.

In this encryption algorithm process, it uses different round keys. The keys are utilized with mathematical operations on an array of data. The data is shown in blocks of specific size. The array is called as state array. This encryption process includes following process:

i. Obtain the different round keys from cipher key.

ii. Initialize the state array with block data or plaintext.

iii. Add round key to start with initial state array.

iv. Perform the process of state manipulation in nine rounds.

v. After tenth round of manipulation, resulting the final output as cipher text.

AES encryption algorithm is used by the U.S. government to protect confidential information. Due to AES flexibility, this encryption algorithm is applicable for hardware and software implementation throughout the world for sensitive data encryption.

2.6.3 Blowfish

Blowfish is a symmetric-key block cipher that was developed by Bruce Schneier in 1993. He designed the Blowfish algorithm and made it accessible for the public to use (Schneier,1993). It was his intent to create new algorithm to serve the world with a new encryption standard. Blowfish encryption algorithm have variable key length from 32 bits to 448 bits. Blowfish encryption algorithm operates on 64 bits block size and it have 16-round Feistel cipher that uses large key dependent S-Boxes. Each S-Box have 32 bits of data.

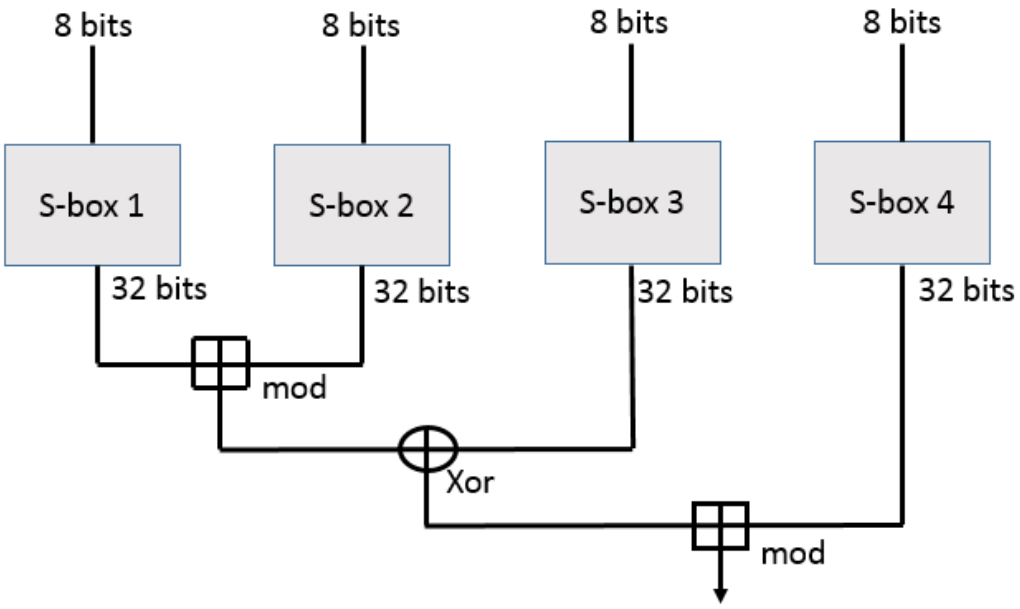

Figure 2.6 Blowfish Function F.

Figure 2.6 shows the Blowfish's F- function that splits the 32 bit input into four 8-bit quarters, and uses the quarters as input to S-boxes. The outputs are added (Mod) modulo 2^32 and XORed to produce the final 32-bit output which are the encrypted data. As for decryption, it occur at another end and the same process takes place, but in reverse and opposite order. Figure 2.7 shows the procedure of Blowfish.

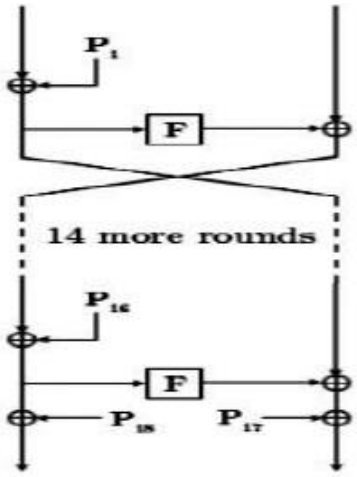

Figure 2.7 Blowfish Procedure

Blowfish algorithm was first introduced in 1993, but until now it has not been cracked yet and no attack has been successfully launched against Blowfish encryption algorithm. This algorithm can be applied in hardware applications, and like most other ciphers, it is also used in software applications because this algorithm implement a good encryption rate in software. It is faster than any other cipher algorithm. In most encryption research, Blowfish encryption algorithm is acknowledge as the best encryption algorithm because the level of security that it offers and the encryption speed, which is better than the most of the encryption algorithm.

2.7 Comparison Between Various Algorithms

The comparison between the three algorithms which are 3DES, AES and Blowfish are summarized as shown in Table 2.2.

Table 2.2 Comparison of Various Algorithms on the basis of Different Parameters

Parameter	**3DES**	**AES**	**Blowfish**
Development	In 1978 by IBM.	In 2001 by Vincent Rijmen and Joan Daeman.	In 1993 by Bruce Schneier.
Key Length (Bits)	168, 112	128, 192, 256	Variable key length i.e. 32 - 448
Rounds	48	10, 12, 14	16
Block Size (Bits)	64	18	64
Attacks Found	Related Key attack.	Key recovery attack and Side channel attack.	No attack is found.
Level of Security	Adequate security	Excellent security	Highly secure
Encryption Speed	Very slow	Faster	Very fast

2.8 Related Work

The security issues in robot operating systems is an active and popular topic for some time. In their work (Bonaci, Yan, Kohno, & Chizeck, 2015) analysed vulnerabilities in the Raven II Surgical Robot. Raven II is a tele-operated robotic system designed to support research in advanced techniques of robot-assisted surgery. It uses open standards software including Linux and ROS. It is a remotely controlled robot. Operators can be nearby or at a completely separate location.

It was found out that there was no authentication and encryption in the communication link. Therefore, authors were able to successfully perform man-in-the-

middle attacks and consequently execute the following intent modification attacks. They were also able to perform hijacking attack. With no authentication in place, the only required attribute in order to take the control of the robot was the sequence number of the packet. After the moderate time of eavesdropping on the network, they were able to find out the current packet sequence number and take full control of the robot by sending any desired command. Obviously all of the performed attacks are unacceptable for a surgical robot and could lead to horrible consequences during the real surgery.

(Denning, Matuszek, Koscher, Smith, & Kohno, 2009) investigated the security of 3 consumer level household robots. Multiple vulnerabilities were discovered. For instance, all communications in some robots were unencrypted and consequently leaked robots authentication credentials and recorded audio/video stream to everyone on the same wireless network. They were also able to control one of the robots with a separately bought off-the-shelf remote control. Authors also expressed a concern regarding robots which have extensive sensing capabilities (audio and video) and the privacy risk they are creating for the environments they are used in. The other concern was regarding robots mobility and ability to grasp objects with actuators and move them or just push objects around in order to deal physical damage.

Based on the related works, we can conclude that ROS communication is not secure because it is implemented in plaintext. The robots are vulnerable to the outsider since they can hack the system easily. Therefore, in this research, cryptography technique which is Advanced Encryption Standard (AES) is used to protect the message-passing distribution between the nodes in the ROS. The message that is passed from the publisher will be encrypted and it will form unmeaning-less text. In order for the subscriber to understand the message, it must have the key to decrypt the message.

CHAPTER 3

METHODOLOGY

3.1 Introduction

In this chapter, methodology that is chosen for this research will be briefly discussed. Methodologies is a technique that are designated to finalized and multi-step ways to system development that will drive the work and influence the quality of the final product of the research project. For better understanding, a process model is used to interpret the progress of the research project.

The hardware and software specification that is used in this research will be outline. The list of hardware and software stated act as an important role to ensure that the research project run very well.

Lastly, a Gantt chart which establish the timeline for the research is covered in this chapter. Gantt chart will remind the researcher throughout the research. Therefore, it can ensure that all the output is delivered within the time limit.

3.2 Research Methodology

Figure 3.1 shows the research methodology. A research project begins with planning, where the details of the project is explained. Scope of the project is defined and method to complete and finish the project is developed in planning phase. The time span for completing the research project according to the distinct number of tasks is discuss in this phase.

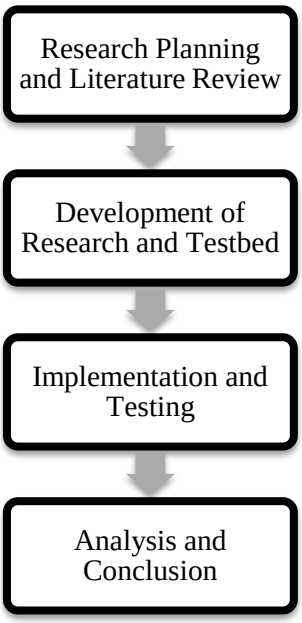

Figure 3.1 Research Methodology

Literature review includes in the first stage where it will provide the clarification about the research project and in what way it was initialized. This stage tends to give an overview of the research study and ensure that the problem is evaluate to accurately develop the models and find the solution that is use in the research project.

Second phase is development of research and testbed where it is the most important stage in the research project. This phase generally involve technical design requirements, such as programming language, data layers, services and more. A design specification will typically be created that outlines how exactly the problem solution that is covered in analysis will be technically implemented.

Third phase is the implementation and experiment stage. Installation and procurement of hardware and software takes place in this stage and both of it must be operational before development of the project begins. Testing is also included in this phase where test run need to be done in the system to remove any bugs. It is crucial phase after implementing the whole project to ensure that the output match the expected results.

Last phase is analysis and conclusion stage where the result is analysed and conclude with a better outcome.

3.3 Research Planning and Literature Review

This phase plays a large role in developing an outstanding and a complete system because it can determine actions that need to be taken and solve the arising problem. It is essential in interpreting the problem and the objective of the research. The objective need to solve the problem statement of project. It is important to ensure that the objective are achieve and the problem is solved.

The thesis is started by analysing the problem, the objective need to be accomplish, limitation, scope as well as the proper platform for development. A schedule is arranged as a manual throughout the project development to keep track the progress and to ensure it can complete within the due date. Gantt chart is used in this phase to plot activity that is involved in the development process. Gantt chart act as a reminder to make sure the process of completing the project followed the timeline.

3.4 Development of Research and Testbed

After analysing the problem of the research, the framework of the design phase is very important in the whole development cycle process. The way of how the algorithm process should be implemented in the system have to be applied in this phase including the tools used in developing the system. The specifications from the first phase are discussed and the system design is prepared for each of the algorithm.

3.4.1 Testbed Description

To classify and evaluate the performance of the encrypted version of ROS communications environment, the following scenario are designed. Figure 3.2 shows a graphical representation of the scenario created.

First, ROS Indigo is installed in the desktop computer to implement the selected algorithm. ROS master is launched in this platform. This enables the nodes to communicate and exchange messages. This platform will act as a talker and have two nodes which are one node publishes the data into */chatter/message* topic and one node connected to this topic that performs data encryption and publishes them into a */chatter/encrypt/message* topic.

Secondly, Raspberry Pi 3B+ is used as “Known Client”. It is installed with ROS Indigo as well and is connected by using Ethernet cable to the network. This client can communicate with the master since it knows the master ROS IP. Decryption node is run in this platform and it will registers to master and subscribes to the topic */chatter/encrypt/messages*. This node decrypts the data that are publish by the talker and display the output.

ROS Master node will keep and store the cryptographic key for encryption and decryption process and it is known by the selected clients only which in this case is the Raspberry Pi. This mechanism can be implemented as public-private key schema as RSA to share the key between the nodes safely.

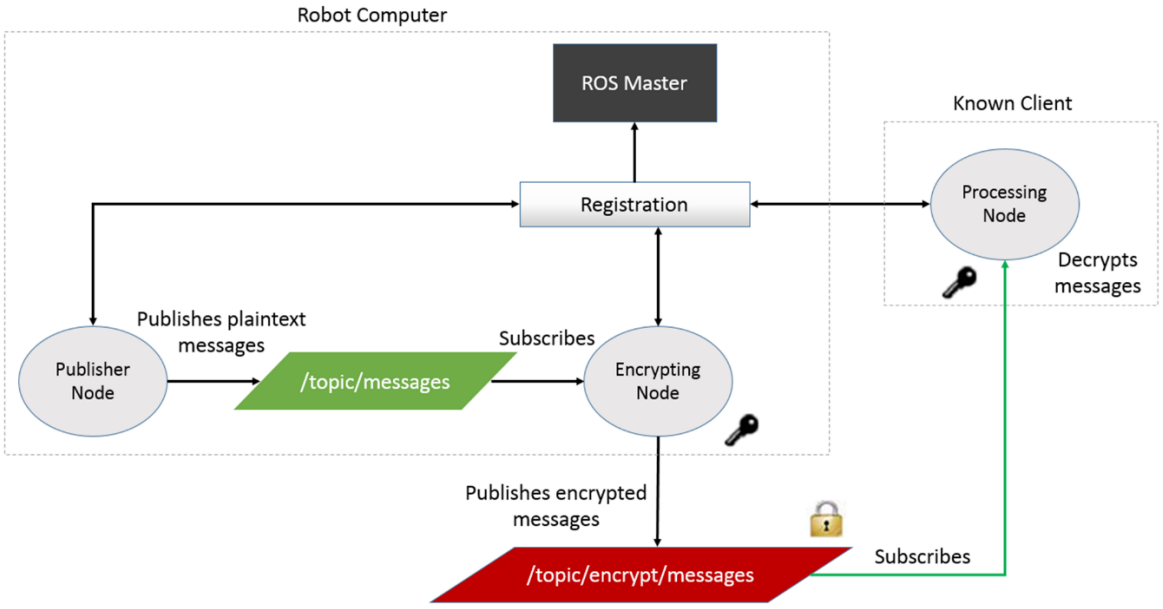

Figure 3.2 Scheme of Scenario for ROS Communications.

3.4.2 Encrypting ROS messages

Recent study (Santos, Pereira, & Couceiro, n.d.) has stated that communication that using cryptography in ROS gives minimal overhead of CPU performance and communication load for systems. The study involve in encrypting data that are conducted in ROS processes using 3DES algorithm as the worse environment to evaluate its behaviour in real environment with ROS. Both encryption and decryption nodes was added in the ROS communication without changing the original message structures for

data sending. The authors use PyCrypto package and evaluated the performance of the system from the computing and communication point of view.

Inspired by the recent study, this research is developed using the similar approach for evaluation. Both of the encryption and decryption nodes is still coded by using Python but AES algorithm is chosen to be implemented in the system. AES is known to be faster than 3DES and it is a symmetric block cipher.

It was stated that symmetric encryption provides better CPU performance rather than asymmetric. Encryption's mode of operation need to be selected to guarantee privacy and authenticity of the message. There are two different modes of operation for cryptography. First mode is represented by the Electronic Code Book mode (ECB) which is a straightforward encryption. Second mode is the modes that use encryption scheme based on Key IV is represented by Cipher Block Chaining (CBC), Cipher Feedback (CFB), Output Feedback (OFB), XEX-based tweaked-codebook mode with cipher-text stealing XTS, and Counter (CTR) modes. These modes are applied to ensure the confidentiality but not message integrity.

In this research, the CBC mode is used because it is easy and require simpler implementation as well as its high level of adoption. CBC mode works by combining plaintext blocks with the previous cipher-text block to encrypt the message. CTR mode is used as alternative which is also widely used. As mentioned by (Krovetz & Rogaway, 2011), both of CBC and CTR modes is identical in terms of encryption and decryption performances when using small blocks, as well as using bigger blocks for decryption process due to its parallelization features.

3.4.3 AES Algorithm

In this research, there are two architectures designed separately and the design mostly followed from a study by (Abdulazeez & Tahir, 2015).

1. The first architecture is designed for AES Encryption Algorithm.

2. The second architecture is designed for AES Decryption Algorithm.

3.4.3.1 Design of AES Encryption Algorithm

The algorithm is design to produce and generate encrypted data (cipher-text) from the plaintext. 'SubShiftbyte' component use one 255-bits Roms named Sbox and also is used by KeyExpanisin component to produce a new key. Rom is functioning to store constant values that cannot be changed. Figure 3.3 shows the stages diagram of AES Encryption.

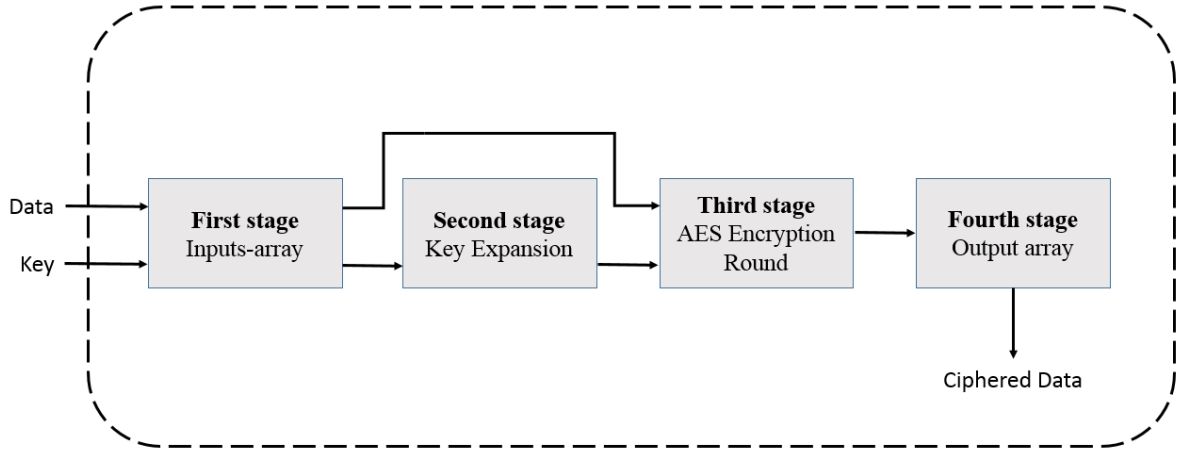

Figure 3.3 The Stages Diagram of AES Encryption

Below are the stages in AES encryption. Assume that 128-bit data and key is used.

1. First Stage

In the first stage, 128-bit data is read and enter in an array called state array. State array represents the plaintext that is processed to produce cipher-text. 128-bit key is also

going through the same process and get into the key array. Both of these arrays have 128 rows and each rows contains of 1-bit.

2. Second Stage

128-bit key array is received from the first stage and is processed to produce keys that will be used in the AES encryption rounds. New key is produced by performing three operations on the key array. Some of the entries in the key array is replaced by other entries that is placed in the ROM memory names Sbox.

During development of new key, new variable named Constant Round will be added. Constant Round's value will keep changing from one round to another. Then, some entries from the key will be XOR-ed with the new entries and Constant Round will produce new key that is specifically for AES encryption round.

3. Third Stage

State array from the first stage is received along with the key array in the previous stage. The state array have some changes since it is going through several components in this stage. Finally, the processed state array from Encryption component is send to the final stage.

As mention before, state array contains changes when going through several components. There are four components, which are Reg_128, SubShiftByte, MixColumn, and AddRoundKey component. All of these components are iterated in 10 rounds except round 10 because MixColumn is not activated.

4. Fourth Stage

New array named output array will receives state array from the previous stage. The content of output array is the same as state array that consists of 128 rows and each rows consist of 1-bit. Cipher-text is represented by output array and the output represents the processed data. Finally, 128-bit cipher-text is produce from 128-bit plaintext. Figure 3.4 shows the graphical representation of AES encryption process.

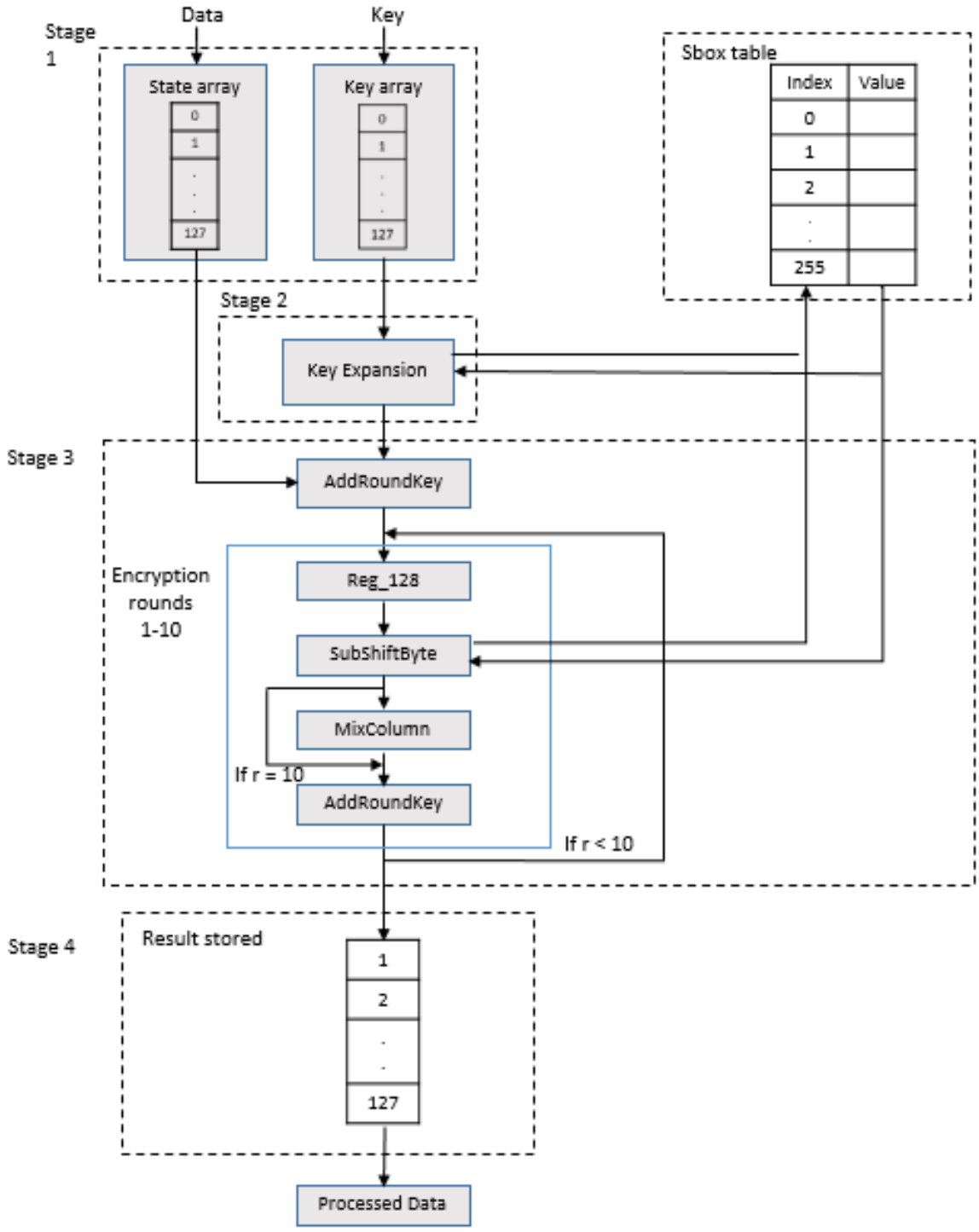

Figure 3.4 AES Encryption Process

3.4.3.2 Design of AES Decryption

The process of decryption is to convert the original data (plaintext) from cipher-text. This process is just the same as encryption algorithm as discussed in section 3.5.3.1 except that the process is reverse. Two (2) 255-bit ROMs is used in AES Decryption algorithm. First ROM is called as Sbox that is used by KeyExpanisin component to generate new key while the second ROM is called as InvSbox is used by InvSubShiftbyte component. Both of the ROMs is functioning to store a constant and fixed value, therefore the value cannot be changed. The block diagram of AES Decryption is shown in Figure 3.5.

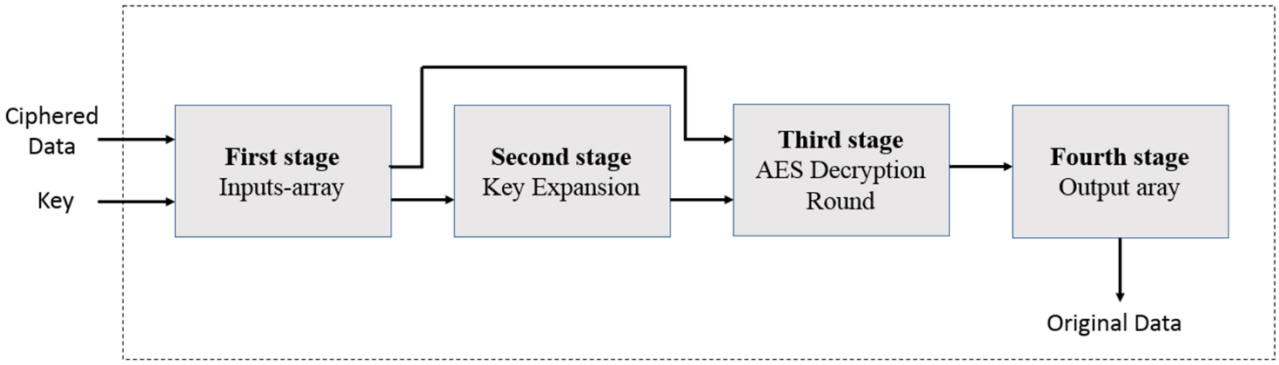

Figure 3.5 The Block Diagram of AES Decryption

Below are the stages in AES decryption. Assume that 128-bit data and key is used.

1. First Stage

128-bit data is read and enter in the state array and 128-bit key is going through the same process and it entering the key array. Both of these arrays contains 128 rows and each rows contains 1-bit. The state array represents the cipher-text that will be processed and converted to get the original data.

2. Second Stage

This stage is the same stage in AES Encryption Diagram, so it is explained in (Sec 3.5.3.1).

3. Third Stage

The third stage receives the state array from the first stage and the key array from the second stage. In this stage, the state array will have changes when going through several components. The state array that already being process from the encryption component is send to the next stage.

As mention before, state array contains changes when going through several components. The components involved are Reg, InvSubShiftByte, InvAddRoundKey and InvMixColumn component. These four (4) components are iterated in 10 rounds except round 10 because InvMixColumn is not activated.

The first component is Register (Reg) which has an input signal 'Reset' for initialization of this register to the value 0x00. InvSubShiftByte component is form by the combination of InvSubByte and InvShiftRow components. InvSubShiftByte component receives state array from Reg component, then it changes the state array elements with other elements that are stored in InvSbox array. Positions of the state array entries is change at the same time to get the correct positions.

A ROM memory is created to store the InvSbox values when the entire state array is changed. InvSbox values are fix and cannot be change. Therefore, InvSubShiftByte component sends the state array which will be the index of the ROM and the ROM returns the values from InvSbox table depends on the index stated.

InvAddRoundKey component will XOR the state array with key array and the result will be stored in the state array. InvMixColumn component functioning to divide the state array into four columns and each of the column is multiplied with the specific row of the array to get a value that is consisting of mixing value from the four elements in the column. The InvMixColumn is active for all rounds in decryption process except the last round which will not be activated.

4. Fourth Stage

The fourth stage receives the state array from the previous stage and enter the new array named output. Figure 3.6 shows the graphical representation of AES decryption process.

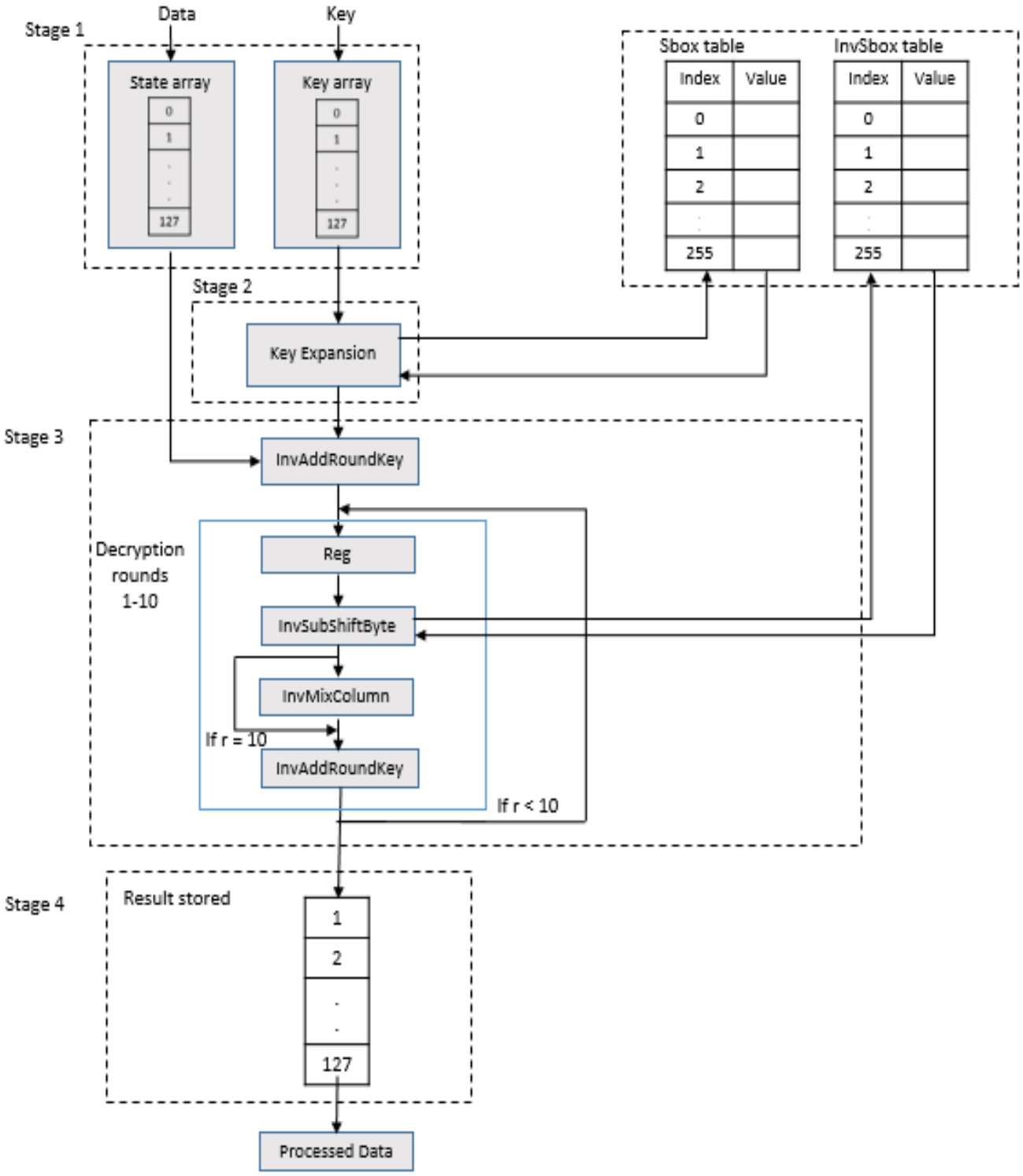

Figure 3.6 AES Decryption Process

3.5 Implementation and Testing

An implementation of the system is made after the design is fully completed. ROS Indigo is used as the framework to complete this research. Results presented in this thesis will show the differences of the performances in message-passing communication between two machines using encryption algorithm. The details about this phase will be discussed in Chapter 4.

The experiment implemented will be tested as all the components are combined. The testing and evaluation is carried out to solve the problem statement and to determine whether the limitation of the existing journals are avoided. The main purpose of this testing is to prove the proposed authentication scheme with real time process in ensuring the accuracy of the result and claims that are made in this research. Moreover, the testing stage allows errors and limitations of the research experiment to be identified so that further improvements can be made to obtain the desire result.

3.6 Hardware and Software Requirement

This section will briefly explain requirements needed in conducting this research on security in Robot Operating System (ROS). It includes software requirements and hardware requirements.

3.6.1 Hardware Requirement

The list of hardware requirement for this research are listed as shown in Table 3.1.

Table 3.1 Hardware requirement

Hardware	Purpose
PC	Act as a talker or publisher for the communication in ROS.
Raspberry Pi 3b+	Act as a listener or subscriber for the communication in ROS.
Ethernet RJ45 Cable	To connect the Raspberry Pi 3b+ to the PC.
ASUS A450L Notebook	To prepare the documentation.
External Hard Disk/Pendrive	A data storage to store the data required for this research.

3.6.2 Software Requirement

Table 3.2 shows software that is used to complete the research. The software is used during documentation and the development of this system.

Table 3.2 Software requirement

Software	Purpose
Microsoft Windows 8 Operating System	As a platform to complete the research.
Microsoft Word 2017	To prepare the documentation.
Gantt Project	To create the Gantt Chart.
Ubuntu 14.04 LTS	As a platform to install ROS Indigo.
ROS Indigo	To implement and test the selected algorithm.
Raspbian Stretch	Software that is required for Raspberry Pi 3b+.

3.7 Gantt Chart

Gantt Chart is important for planning and scheduling the time taken for the research project. It help to assess how long a project should take, determine the sources needed and plan the order of which task should be done first. Refer Appendix A.

3.8 Summary

As a conclusion, this chapter explain about the methodology used to develop the Security of Robot Operating System (ROS) by using Advanced Encryption Standard (AES). Waterfall has been chosen for the methodology because it is the most suitable method that can be used for this research. The design for each algorithm which is 3DES, AES and Blowfish are discussed in this chapter. The software and hardware also has been analysed. The chosen software and hardware is important so that the process of completing this research project can run smoothly.

CHAPTER 4

RESULT AND DISCUSSION

4.1 Introduction

The purpose of this chapter is to discuss in details on how the process of encryption algorithm which is AES is implemented in ROS. To relate to the methodology used, this process occurs during the designing phases. All of the implementation of algorithm and coding will be applied in order to achieve the main goal and objective of this research. The comparison between communication of the ROS nodes before and after encryption algorithm is analysed. At the end of this chapter, the overall findings are summarized.

4.2 Implementation

Implementation is the most essential phase in order to achieve the objective during developing this research. In this section, the security initiatives that is described in previous chapter is tested which is by using Advanced Encryption Standard (AES).

4.2.1 Hardware/Software Set-up

For this research, Intel Core i5 CPU with 7.8 GB of RAM Memory and running Ubuntu 14.04 LTS Operating System is used. There is more than one ROS distribution supported at a time. In this research, ROS Indigo Igloo is used since it is primarily targeted at the Ubuntu 14.04 LTS release. The latest product of Raspberry Pi 3 Model B+ with 1.4GHz 64-bit quad-core processor is used as the subscriber where the messages will be send from the publisher. ROS Indigo is also installed in the Raspberry Pi. Raspbian Stretch is the official operating system for Raspberry Pi and it comes with

plenty of software for education, programming and general use. Raspberry Pi is connected to the PC using RJ45 cable.

4.2.2 Connection of PC and Raspberry Pi

To access the command line of the Raspberry Pi remotely from the PC, Secure Shell (SSH) can be use. Before getting started, make sure the PC and Raspberry Pi is properly set up and connected. In this research, RJ45 cable is used. The first step is to identify IP address of the Raspberry Pi in order to connect to it later. 'ifconfig' command will display the current network status including the IP address. In this case, the IP address for the Raspberry Pi is fe80::6b7b:271b:d58d:fb4. Therefore, command 'ssh -6 pi@ fe80::6b7b:271b:d58d:fb4%eth0' is used to display the command line for Raspberry Pi.

As mention before, the PC will act as a talker while the Raspberry Pi will act as a listener. ROS Master is run in the PC using different terminal and all nodes must be able to configured using the same master via ROS_MASTER_URI. Each machine must advertise itself by a name that all other machines can resolve. In this case, IP address of the both machine is used to ensure that they recognize each other. Table 4.1 shows the IP address and commands that will be used to connect the two machines.

Table 4.1 IP address and Commands for PC and Raspberry Pi

	PC	**Raspberry Pi**
IP	10.42.0.1	10.42.0.97
Command	$ roscore $ export ROS_IP = 10.42.0.1	$ export ROS_MASTER_URI = http://10.42.0.1:11311 $ export ROS_IP = 10.42.0.97

4.2.3 Test : HelloWorld Talker-Listener Node

Noted that in this research, programming language Python is used instead of C++. This is because Python is much simpler, which leads to faster development and great for rapid testing and exploration. In this test, the talker/listener tutorial that are proposed by ROS is used ("ROS_Tutorials_WritingPublisherSubscriber(python) - ROS Wiki," n.d.). This example is already installed in ROS.

This package distributed by ROS as a demo, presents a simple ROS package that create two rospy nodes which is talker and listener. Talker node will publish a Hello World with Timestamp message on “chatter” topic while listener node will subscribes and print the message that is received from the talker node. Both of the nodes can be found in rospy_tutorial package. Talker.py and listener.py script are downloaded from the tutorials. Figure 4.1 shows the source code for talker.py. Refer to the line number 7, it declaring the node to publish the message to ‘chatter’ topic using message type ‘String’. The ‘queue_size’ limits the amount of queued messages is the listener did not receive any of the messages fast enough. The next line is very crucial where it tells the rospy the name of the node which is talker. The ‘rate’ code is to ensure that the message go through the loop 10 times per second.

```
1  #!/usr/bin/env python
2  # license removed for brevity
3  import rospy
4  from std_msgs.msg import String
5
6  def talker():
7      pub = rospy.Publisher('chatter', String, queue_size=10)
8      rospy.init_node('talker', anonymous=True)
9      rate = rospy.Rate(10) # 10hz
10     while not rospy.is_shutdown():
11         hello_str = "hello world %s" % rospy.get_time()
12         rospy.loginfo(hello_str)
13         pub.publish(hello_str)
14         rate.sleep()
15
16 if __name__ == '__main__':
17     try:
18         talker()
19     except rospy.ROSInterruptException:
20         pass
```

Figure 4.1 Source Code for talker.py

Figure 4.2 shows the source code for listener.py. The source code for listener.py is similar to talker.py except that it apply a new callback-based mechanism to subscribe the messages.

```
#!/usr/bin/env python
import rospy
from std_msgs.msg import String

def callback(data):
    rospy.loginfo(rospy.get_caller_id() + "I heard %s", data.data)

def listener():

    # In ROS, nodes are uniquely named. If two nodes with the same
    # node are launched, the previous one is kicked off. The
    # anonymous=True flag means that rospy will choose a unique
    # name for our 'listener' node so that multiple listeners can
    # run simultaneously.
    rospy.init_node('listener', anonymous=True)

    rospy.Subscriber("chatter", String, callback)

    # spin() simply keeps python from exiting until this node is stopped
    rospy.spin()

if __name__ == '__main__':
    listener()
```

Figure 4.2 Source Code for listener.py

After the communication between the talker and listener is successfully done, AES algorithm is applied in the source code. In this research, PyCrypto package is used. It is an extended python Cryptography Toolkit that can simplify the method to encrypt and decrypt the data. Encryption will take place in talker nodes while Decryption will take place in listener node. Figure 4.3 shows the AES encryption and decryption algorithm.

The counter string value with length of 16 bytes and 16 bytes length for the key size is used. Note that AES keys may have 128 bits (16 bytes), 192 bits (24 bytes) or 256 bits (32 bytes) long. Encryption object is created by passing the key, AES mode and the counter value. Counter is required to be sent as a callable object hence ‘lambda’ is used as an anonymous function where the function in not bound to a name. The decryption algorithm is almost similar to the encryption algorithm.

This research were recorded using rosbags, which is a file format in ROS that is functioning to store ROS message data. It is typically created by running ‘rosbag record –a’ command tool when the nodes is subscribing to the topics and it will store the information regarding the message passing distribution that happened in ROS

automatically. The rosbag can display the information includes duration, size of the message, number of message that was send or receive, topics and nodes that are involved in the communication by running ‘rosbag info’ command line.

```
39  import rospy
40  from std_msgs.msg import String
41
42  from Crypto.Cipher import AES
43
44  counter = "H"*16
45  key = "H"*16
46
47  def encrypt(message):
48      encrypto = AES.new(key, AES.MODE_CTR, counter=lambda: counter)
49      return encrypto.encrypt(message)
50
51  def decrypt(message):
52      decrypto = AES.new(key, AES.MODE_CTR, counter=lambda: counter)
53      return decrypto.decrypt(message)
54
```

Figure 4.3 AES Encryption and Decryption Algorithm

4.3 Evaluation Parameters

Each of the encryption technique has its own strong and weak characteristics. In this research, AES with different key length, which are 128 bits, 192 bits and 256 bits, will be evaluated based on several features. Analysis of the algorithm is done by using the following metrics and parameters:

- **Encryption Time**

The encryption time is the time taken to convert plain text to cipher-text. It depends on the key size, block size and also the mode used for the encryption algorithm. In this research, the encryption time is measured in seconds. Encryption time can affect the performance of the communication in the system.

- **Decryption Time**

Decryption time refer to the time needed for the system to get back the original text or plain text. Decryption time is almost the same with encryption time to ensure that the algorithm is fast and responsive. Decryption time can give impacts to the system and it is measured in seconds.

- **Real Time**

It is a wall clock time where the time is taken from start to finish of the call. The running publisher/talker and subscriber/listener node is performed between 40 to 45 seconds time lapse.

- **Time user**

Amount of CPU time spent in user mode code that is outside the operating system within the process. This is actual CPU time used in executing the process. Other process and time the process spends blocked is not counted in this parameter. In a simplest way to understand, time user is the time that the program took to execute the process.

- **Time Sys**

Amount of CPU time spent in the operating system within the process. The executed CPU time spent in the system calls within the operating system, as opposed to library code, which is still running in user-space. Similar to user, this is only CPU time used by the process.

- **Number of Messages**

Number of messages that is executed by publisher and message that can be subscribes by the subscribers is analysed by using plaintext version, encrypted and decrypted version.

4.4 Result and Discussion

This section describes the results when the nodes is running and encryption algorithm is applied. The time lapse is taken manually within 40 to 45 seconds and both of listener and talker nodes are terminate by pressing Ctrl-C. By using the Unix command time, the result of the launching ROS nodes which are talker and listener can be analysed.

Figure 4.4 present the output from talker and listener nodes without using any encryption algorithm. Publisher/talker node (top window) publish stream of “Hello World” with timestamp messages 10 times per second and sending them to any nodes that are listening. In this case, Subscriber/listener node (bottom window) is run and print out the “Hello World” message that is subscribed from the “chatter” topic.

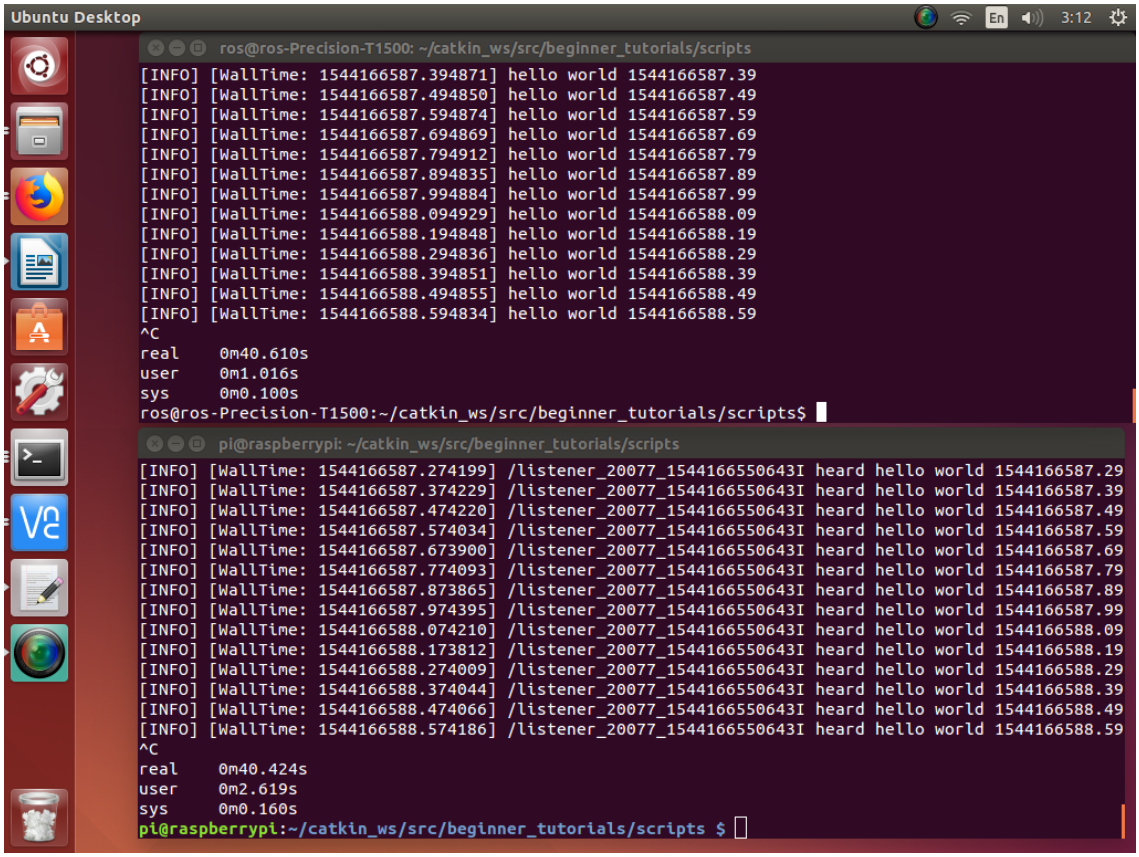

Figure 4.4 Output From Publisher/Talker and Subscriber/Listener Nodes Before Encryption.

While leaving the three of the terminal (roscore, talker and listener) open and running, new terminal is open to be able to view the scenario for the experimental setup. Command 'rqt_graph' represent the current status and state of the system. Figure 4.5 shows the ROS setup running in each iteration.

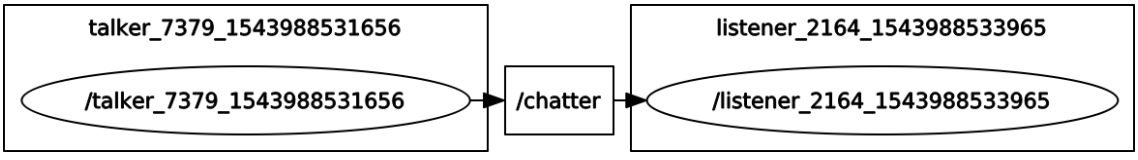

Figure 4.5 Visual Representation Using Rqt_Graph for the Communication of the Talker and Listener Nodes

After implementing AES algorithm in the ROS communication, the message sent will different from the plaintext since it is encrypted and form cipher text. Figure 4.6 shows a screenshot of applying AES-128 algorithm in both of the nodes. The top window is talker nodes terminal where encryption algorithm is applied and it form un-meaningful text instead of "Hello World" that is shown in Figure 4.4. The below window represent the listener node where decryption algorithm is inserted for the subscriber to decipher the message and "Hello World" is shown.

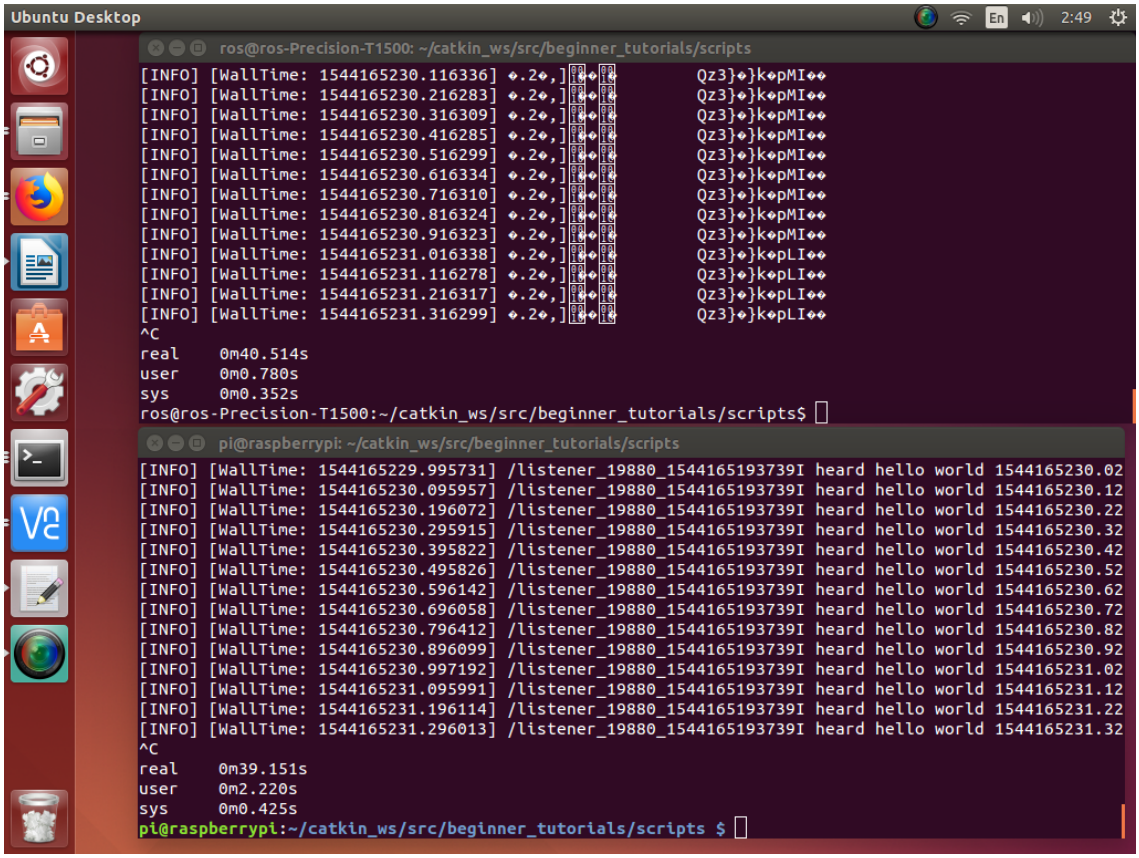

Figure 4.6 Output From Publisher/talker and Subscriber/listener Nodes After Implementing AES-128 algorithm.

Table 4.2 presents the CPU time used when both of the nodes running in different machines. As mention before, Unix command time is used to determine the time of CPU needed to execute the ROS nodes. The parameter is evaluated by using plaintext and three different key length of AES, which are AES-128 (16 bytes), AES-192 (24 bytes), and AES-256 (32 bytes).

Firstly, the test for plaintext publisher/talker is performed for a time lapse within 40 seconds, CPU time is found to be 1.116 seconds. It was a user time of 1.016 seconds and a sys time 0.100 seconds. As for the subscriber/talker, the total of CPU time is 2.779 seconds. The user time is 2.619 seconds and a sys time of 0.160 seconds.

Table 4.2 Time in seconds of CPU spent and number of messages

	Plain		**AES-128**		**AES-192**		**AES-256**	
	Talker	**Listener**	**Talker**	**Listener**	**Talker**	**Listener**	**Talker**	**Listener**
Time Running	40.610	40.424	40.514	39.151	41.024	40.468	41.606	42.320
Time User	1.016	2.619	0.780	2.220	1.032	2.673	1.192	2.481
Time Sys	0.100	0.160	0.352	0.425	0.196	0.069	0.088	0.290
Total CPU	1.116	2.779	1.132	2.645	1.228	2.742	1.280	2.771
Messages	1958	1180	1950	1175	1963	1179	2019	1217

When AES algorithm is applied, it is possible to see that encryption process consumes more CPU than the plaintext version process. For example, AES-128 algorithm is running for 40.514 seconds and a total time for CPU is 1.132 seconds. This shows that the total CPU time is increases to 0.016 seconds. The percentage differences is even higher in encryption of AES-192 and AES-256. It is proven that the encryption and decryption method consumes more CPU than plain text as shown in Figure 4.7. The result demonstrate that AES with 128 bits key length takes less time to encrypt and decrypt the message because it only have 10 rounds. AES-192 and AES-256 takes more time to process the algorithm in the system because they have more rounds compared to AES-128 which are 12 rounds and 14 rounds respectively.

In addition, number of messages that are executed also can be seen in rosbags during the launching of the nodes. AES-256 shows that it have the highest number of messages encrypted which is 1973 messages and AES-128 have the smallest number of messages encrypted compared to AES-192.

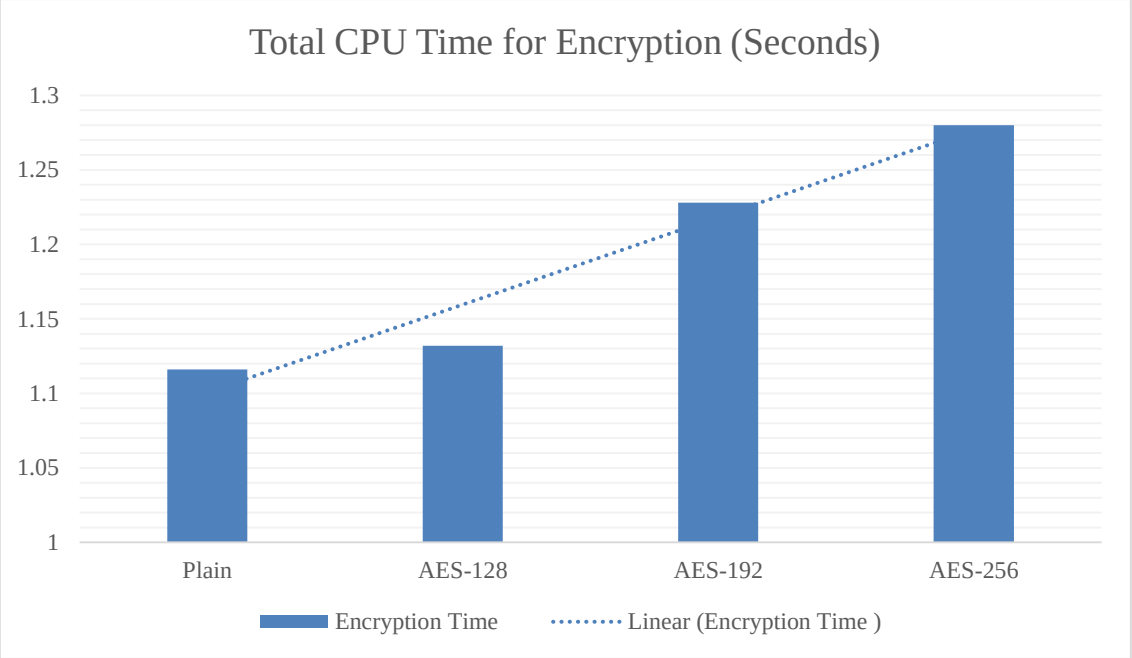

Figure 4.7 Total CPU Time for Encryption In Seconds

In the end, the overall result of encryption algorithms are presented based on different evaluation parameters. All cryptographic algorithm depends on their own characteristics includes block size, length of the key and number of rounds. As for AES, the number of rounds is depends on the length of the key. 128 bits, 192 bits and 256 bits of key length will have 10, 12 and 14 rounds in the algorithm. In order to apply the correct algorithm according to the applications, the strength, weakness and performance must be evaluated based on different parameters.

Although the plaintext version is faster than encryption algorithm, it is not secure in the ROS system communication since the third party or outsider can easily read and manipulate the data that is send from publisher to subscriber. From the above result, we can see that the larger the length of the key size, the more rounds is needed and more time it took to encrypt and decrypt the data. Therefore, it may be more difficult to analyse, require more constants and code where can be more complex. Hence, it may be good for the security of the message-passing distribution in ROS communication.

CHAPTER 5

CONCLUSION

5.1 Introduction

In the early stage of this research, the project scopes with problem statements are identified. The objective of this research will be deliberate whether it is achieve or not. The overall discussion will be concluded in this chapter.

5.2 Conclusion

In this research, three objectives is used as a reference to solve the problem statement. The first objective is to identify the possible security attack that happen in the robot operating system (ROS). Unauthorized publishing, unauthorized data access and Denial of Service (DoS) attack are the examples of security attack that can be carried out by the attacker to disturb the communication between the launching nodes in ROS.

The second objective is to implement cryptography technique which is Advanced Encryption Standard (AES). AES is symmetric block cipher that have variable key length of 128, 192 and 256 bits. Each cipher able to encrypt and decrypt messages in 10, 12 and 14 rounds depends on the key size. Encryption algorithm is implemented in Publisher/Talker node where it will publish messages to Subscriber/Listener node. Subscriber/Listener node must have decryption algorithm for the node to be able to read the original message.

The third objective is to evaluate the performance of AES algorithm based on the system parameter in ROS. The time of encryption and decryption of the AES algorithm is identified by using the CPU total time. The number of messages that was able to

execute by both of the nodes is also presented by using rosbags. From the results, it shows that the longer the length of the key, more rounds are required. Therefore, longer time is taken to complete the process of encryption and decryption.

5.3 Limitation

The limitations of the research is observe during evaluating and the whole process of completing the research project. This research used the latest version of Raspberry Pi which is model 3b+ and it requires its own operating system which is Raspbian Stretch. It takes time to install ROS Indigo in Raspberry Pi since the installation command in the manual provided is not compatible with the latest version of the software.

5.4 Future Work

There are few enhancements that can be done to improve the research study of the ROS communication between publisher and subscriber. In this research, the time 40 seconds is set manually. Therefore, the time can be set by using python time code in the future work to get a more accurate results. A proposed direction for the future work could be to analyse the performance or security in greater depth. For instance, the algorithm can be analysed and test using variation file size to identify the time CPU needed to execute the encrypted message.

AES algorithm also can be further explore by using different modes such as CBC, CFB and OFB. Each mode of the operation has its own parameters which are important to provide the necessary security in the Robot Operating System (ROS). This algorithm also could be implement to any other machine such as Arduino where the encryption and decryption performance between those two machines can be compare.

REFERENCES

Abdulazeez, A. M., & Tahir, A. S. (2015). Design and Implementation of Advanced Encryption Standard Security Algorithm using FPGA, (September 2013).

Adenowo, A. A. A., & Adenowo, B. A. (2013). Software Engineering Methodologies: A Review of the Waterfall Model and Object-Oriented Approach. *International Journal of Scientific & Engineering Research*, *4*(7), 427–434. Retrieved from http://www.ijser.org/researchpaper%5CSoftware-Engineering-Methodologies-A-Review-of-the-Waterfall-Model-and-ObjectOriented-Approach.pdf

Bhanot, R., & Hans, R. (2015). A Review and Comparative Analysis of Various Encryption Algorithms, *9*(4), 289–306.

Bonaci, T., Yan, J., Kohno, T., & Chizeck, H. J. (2015). To Make a Robot Secure: An Experimental Analysis of Cyber Security Threats Against Teleoperated Surgical Robots, (April).

Denning, T., Matuszek, C., Koscher, K., Smith, J. R., & Kohno, T. (2009). A Spotlight on Security and Privacy Risks with Future Household Robots : Attacks and Lessons, (October 2014). https://doi.org/10.1145/1620545.1620564

Dorsey, D. W., Martin, J., Howard, D. J., & Coovert, M. D. (2017). Cybersecurity issues in selection. *Handbook of Employee Selection, Second Edition*, 913–929. https://doi.org/10.4324/9781315690193

Hwang, M., & Liu, C. (2005). Authenticated Encryption Schemes : Current Status and Key Issues, *1*(2), 61–73.

Krovetz, T., & Rogaway, P. (2011). The Software Performance of Authenticated-Encryption Modes, *2011*(Fse), 1–24.

Morante, S., Victores, J. G., & Balaguer, C. (2015). Cryptobotics : why robots need cyber safety, *2*(September), 23–26. https://doi.org/10.3389/frobt.2015.00023

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., … Mg, A. (2009). ROS: an open-source Robot Operating System. *Icra*, *3*(Figure 1), 5. https://doi.org/http://www.willowgarage.com/papers/ros-open-source-robot-operating-system

Rodr, F. J., Casado, F., Fern, C., & Mart, F. (2016). Cybersecurity in Autonomous Systems : Evaluating the performance of hardening ROS, (June), 47–53.

ROS_Introduction - ROS Wiki. (n.d.).

ROS_Tutorials_WritingPublisherSubscriber(python) - ROS Wiki. (n.d.).

Santos, M. A., Pereira, S., & Couceiro, M. S. (n.d.). On the Security of Robotic Applications Using ROS, 273–289.

Shirabadagi, S. S., & Nadagoud, S. (2017). A new encryption methodology of aes algorithm using high speed s-box, *4*(7), 37–42.

Stallings, W. (n.d.). *D ATA AND C OMPUTER*.

APPENDIX A
GANTT CHART

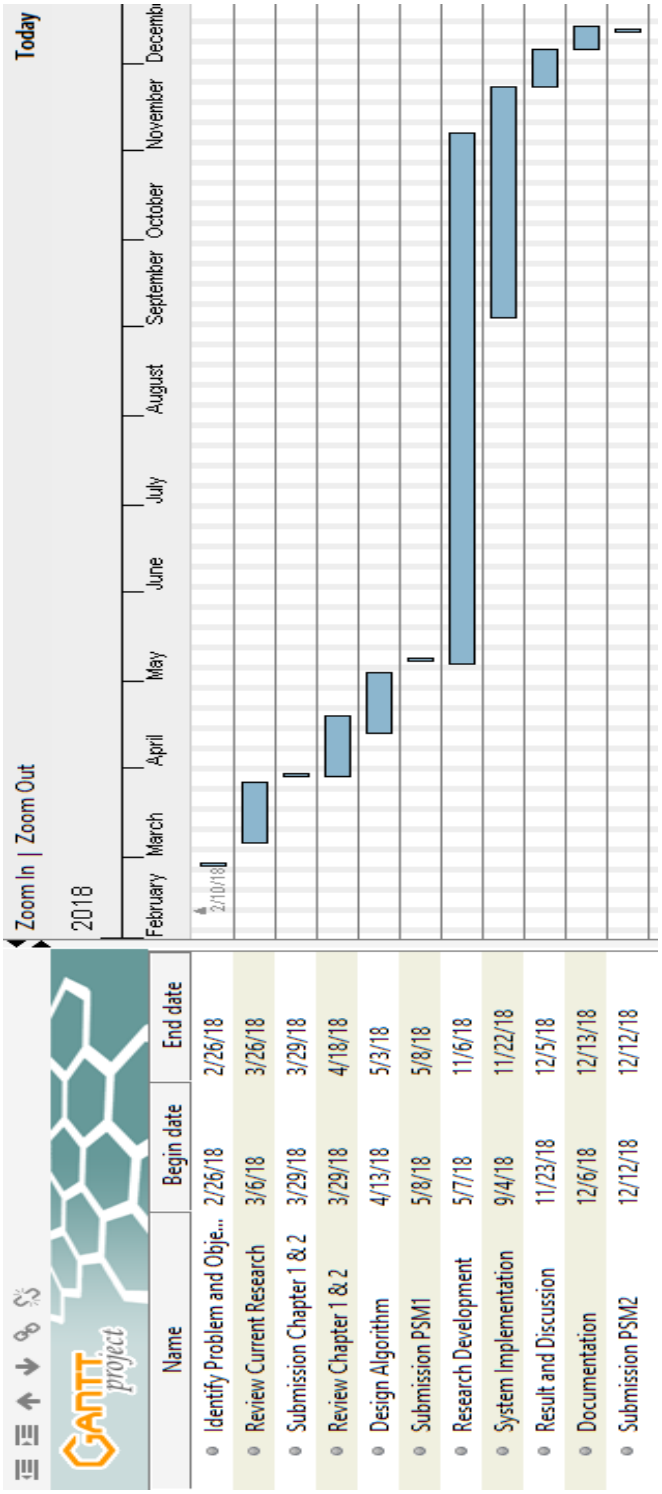

Figure A.1 Gantt Chart

APPENDIX B

EXPERIMENTAL RESULT

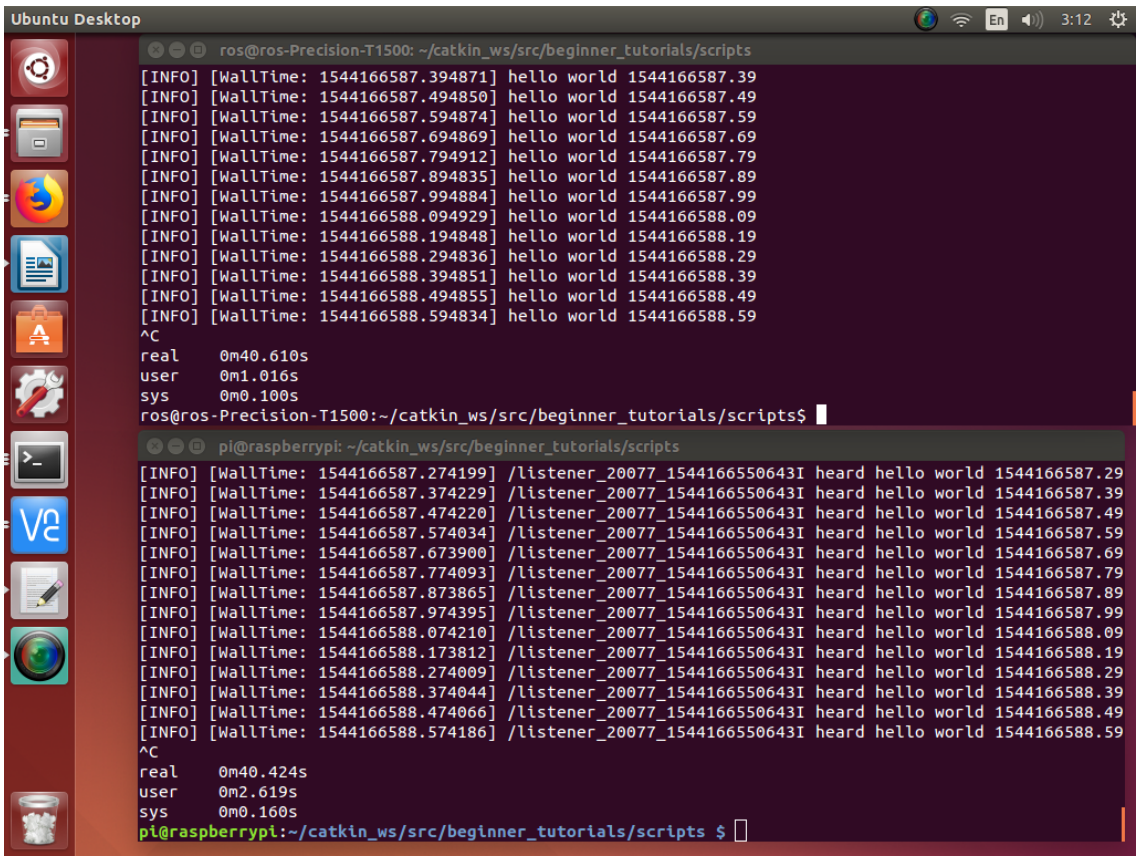

Figure B.1 Results for Plaintext Version

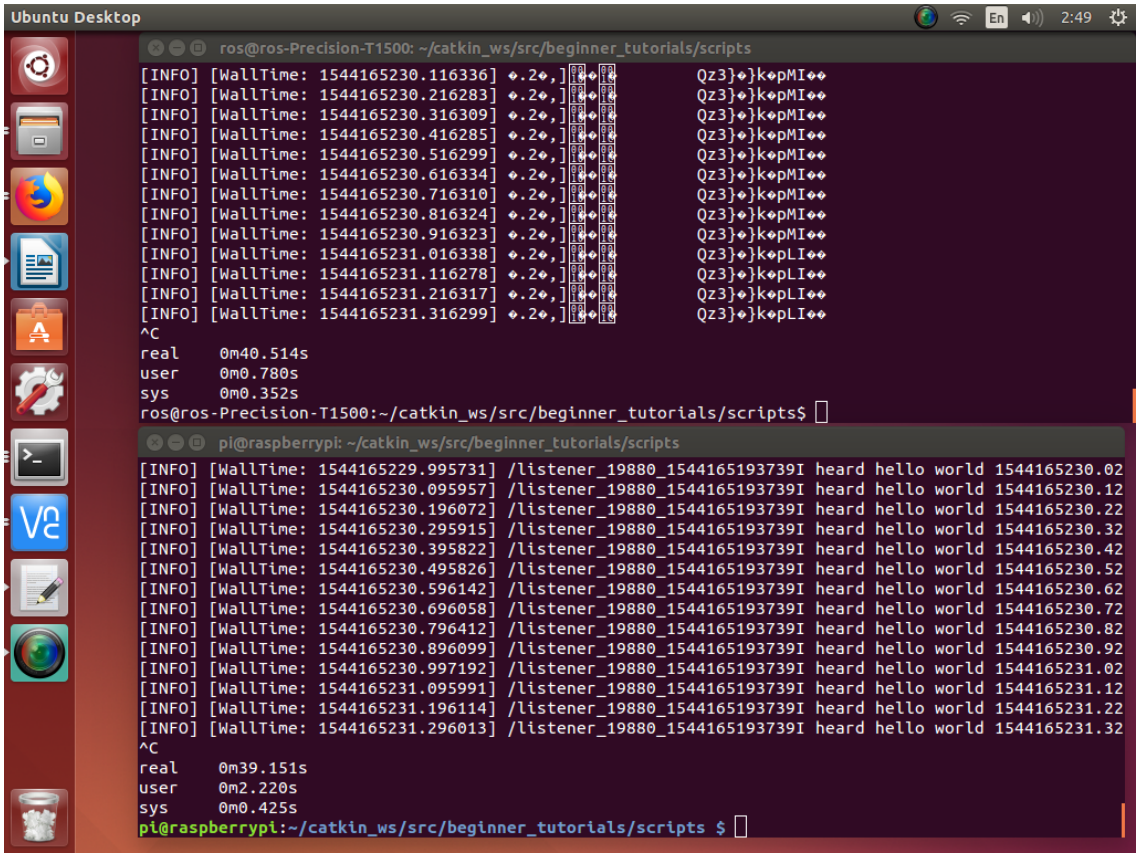

Figure B.2 Result for AES-128

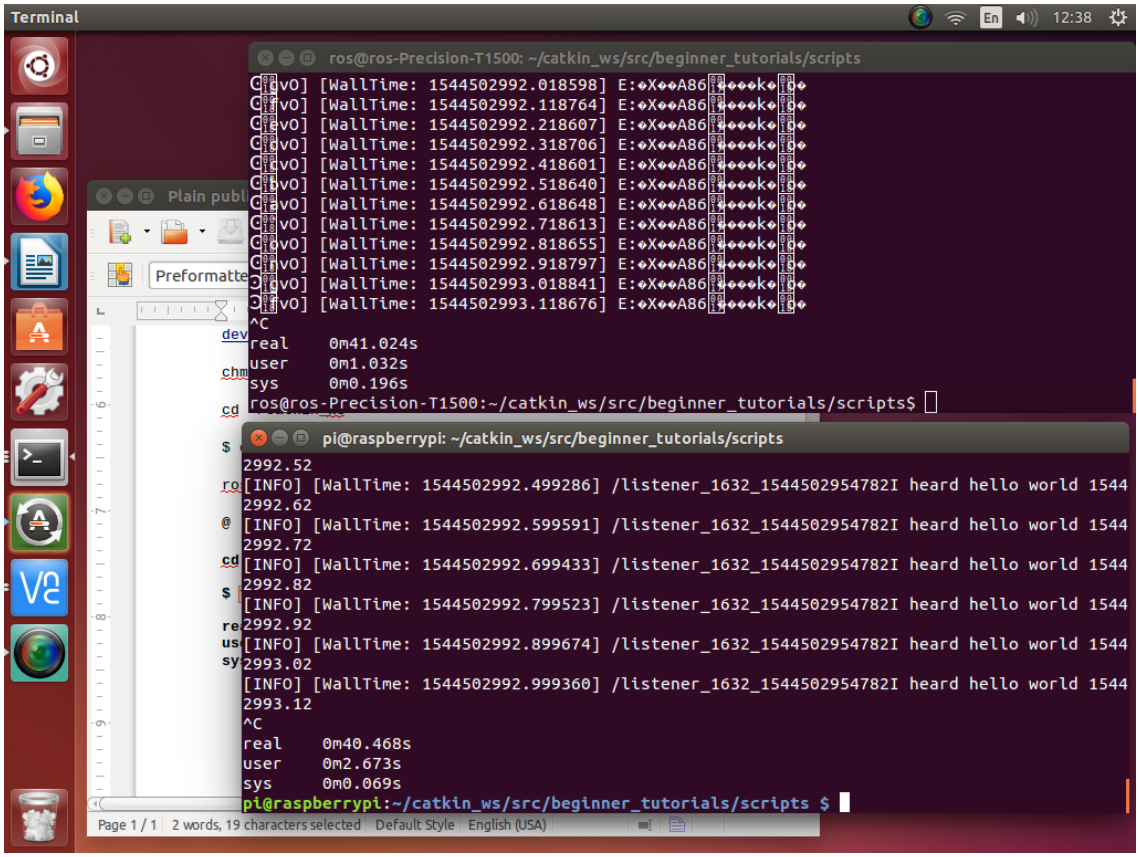

Figure B.3 Results for AES-192

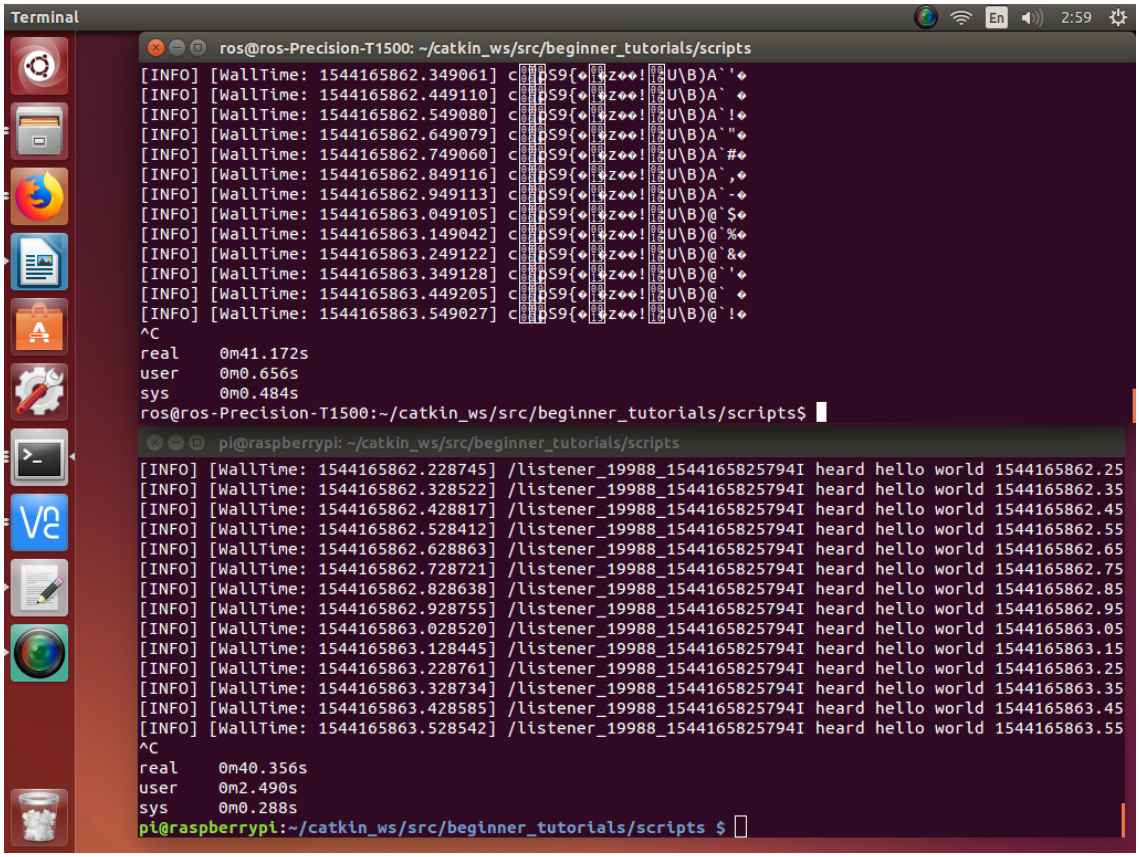

Figure B.4 Results for AES-256