TELKOMNIKA Indonesian Journal of Electrical Engineering
Vol. 15, No. 3, September 2015, pp. 526 ~ 533
DOI: 10.11591/telkomnika.v15i3.8423

Secured Communication Among HMI and Controller using RC-4 Algorithm and Raspberry Pi

Albert Sagala*, Deni Lumbantoruan, Epelin Manurung, Iroma Situmorang, Adi Gunawan
Cyber Security Research Centre (CSRC) - Del Institute of Technology,
Jl.Sisimangaraja Desa Sitoluama Kecamatan Laguboti
*Corresponding author, e-mail: albert@del.ac.id, toruan@del.ac.id

Abstract

Supervisory Control and Data Acquisition is a technology applied to be able to perform supervison, controlling and acquisition data remotely. SCADA technology utilization had a positive impact on the efficiency of an industrial plant to produce the desired product. Daily data containing information from the sensors and actuators installed on the plant in realtime allows decisions to be made as early as possible in order to obtain a quality product. In the beginning the implementation of a plant with SCADA technology, the network is formed isolated from the outside network (LAN or Internet). So it can be ascertained that the communication that occurs on the SCADA network is safe from the threat of crackers. In fact, SCADA network allows it to be connected to the Internet, so that the data of the plant can be monitored via the Internet, so the information about the state of the plant can be monitored in realtime and can be taken quickly if it is known there are anomalies on the control system. In the research we designed a method of encryption and decryption against the lines of communication between the HMI (Human Machine Interface) and the Controller on an industrial minipant contained in Lab CSRC IT Del. Raspberry Pi is used as a gateway between the HMI and the Controller. While Algorithm RC 4 are used as the algorithm for encrypting data between the HMI and the Controller. In the results, we can use the Rapsberry Pi to secure the communication between the HMI and Controller.

Keywords: *secured SCADA, encryption RC-4 for SCADA, rapsberry Pi gateway*

1. Introduction

SCADA is a combination of automation technologies that support the supervision, controlling and monitoring sensors and actuators in the plant. SCADA was built using client-server architecture, SCADA controller as a client and Human Machine Interface (HM) as a server [1]. SCADA systems represent an important tool that enables a quick and precision data acquisition; certainly they are a perfect source for Big Data, since the quantity of data they generate is much larger than any other systems dealing with information, like for corporate systems.

SCADA is considered to be a backbone not only for the plant (organization), but for the country itself. The incredibly high number of SCADA systems operating worldwide, more than 3 million, and the fact that they are implemented in many sensitive and strategic industries, makes us seriously take into consideration security requirements. Cyber attack to Criticial Infrastructure is also increase with level of destruction also increasing [1, 2].

The migration to TCP/IP networks and using commoditiy platform for Modbus clients make these systems potentially vulnerable to attacks againts the operating system and the network, even if such attacks are not against the Modbus protocol. Modbus clients today typically run a version of the Windows operating sytem. For some application environments in which it is expensive or difficult to shut down the control system, operating system pathes may not be up to date. So these systems may run with known vulnerabilities.

Information was transmitted plaintext between the HMI and SCADA. So, data will be easily seen or interpret by the cracker / hacker who entered into the SCADA network through the stages of ethical hacking [3-5]. Solution that can be applied to secure SCADA communication lines can be done several ways, for example by VPN technology if the data is passed via the public internet communication. Another way is by adding a gateway that will perform the encryption and decryption on data transmitted or received.

Received May 25, 2015; Revised July 28, 2015; Accepted August 14, 2015

In our research, we use Raspberri Pi as a gateway between the controller and HMI. The raspberry will have a function to decrypt the message sending from HMI and encrypt the message when send the data from controller to HMI. So that, the data transmitted is encrypted and make difficult to attacker to get the message.

1.1. Modbus TCP/IP

Modbus is a serial communications protocol originally published by Modicon (now Schneider Electric) in 1979 for use with its programmable logic controllers (PLCs). Simple and robust, it has since become a standard communication protocol, and it is now a commonly available means of connecting industrial electronic devices [6].

Modbus TCP/IP is a variant of the Modbus family of simple, vendor-neutral communication protocols intended for supervision and control of automation equipment. Specifically, it covers the use of modbus messaging in an ‘Intranet’ or ‘Internet’ environment using the TCP/IP protocols. The most common use of the protocols at this time is for Ethernet attachment of PLC’s, I/O modules, and ‘gateways’ to other simple field buses or I/O networks. In Modbus, data transactions are traditionally stateless, making them highly resistant to disruption from noise and yet requiring minimal recovery information to be maintained at either end.

Programming operations, on the other hand, expect a connection-oriented approach. This was achieved on the simpler variants by an exclusive ‘login’ token and on the Modbus Plus variant by explicit ‘Program Path’ capabilities which maintained a duplex association until explicitly broken down.

Modbus TCP/IP handles both situations. A connection is easily recognized at the protocol level, and a single connection may carry multiple independent transactions. In addition, TCP/IP allows a very large number of concurrent connections, so in most cases it is the choice of the initiator whether to reconnect as required or re-use a long-lived connection.

In practice, Modbus TCP embeds a standard Modbus data frame into a TCP frame, without the Modbus checksum, as shown in the following diagram.

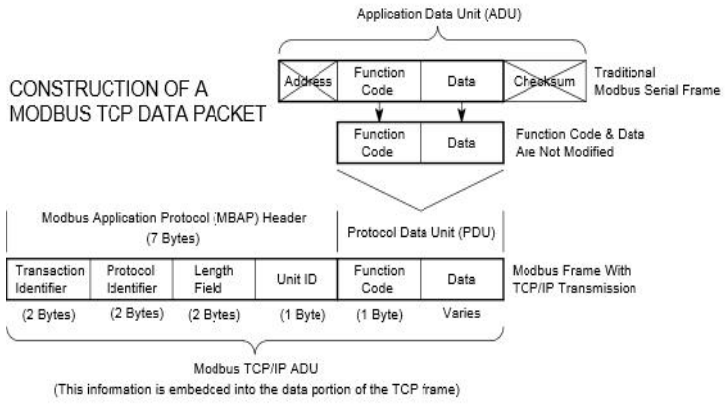

Application Data Unit (ADU)
CONSTRUCTION OF A MODBUS TCP DATA PACKET
Address
Function Code
Data
Checksum
Traditional Modbus Serial Frame
Function Code
Data
Function Code & Data Are Not Modified
Modbus Application Protocol (MBAP) Header (7 Bytes)
Protocol Data Unit (PDU)
Transaction Identifier
Protocol Identifier
Length Field
Unit ID
Function Code
Data
Modbus Frame With TCP/IP Transmission
(2 Bytes)
(2 Bytes)
(2 Bytes)
(1 Byte)
(1 Byte)
Varies
Modbus TCP/IP ADU
(This information is embedded into the data portion of the TCP frame)

Figure 1. Standard Modbus Data Frame

We know that that Modbus operates according to the common client/server (master/slave) model. That is, the client (master) sends a request telegram (service request) to the server (slave), and the server replies with a response telegram. If the server cannot process a request, it will instead return an error function code (exception response) that is the original function code plus 80H (i.e. with its most significant bit set to 1)

Modbus data model has a simple structure that only differentiates between four basic data types as show on Table 1.

Table 1. Modbus Data Model

input discretes	single bit, provided by an I/O system, read-only
output discretes	single bit, alterable by an application program, read-write
input registers	16-bit quantity, provided by an I/O system, read-only
output registers	16-bit quantity, alterable by an application program, read-write

1.2. Raspberry Pi

The Raspberry Pi is a series of credit card-sized single-board computers developed in the UK by the Raspberry Pi Foundation with the intention of promoting the teaching of basic computer science in schools [7].

The original Raspberry Pi is based on the Broadcom BCM2835 system on a chip (SoC), which includes an ARM1176JZF-S 700 MHz processor, VideoCore IV GPU, and was originally shipped with 256 megabytes of RAM, later upgraded (models B and B+) to 512 MB. The system has Secure Digital (SD) (models A and B) or MicroSD (models A+ and B+) sockets for boot media and persistent storage.

Figure 2. Rapsberry Pi 2

In this paper the research method is first addressed in Section 2. Then, in Section 3 the experimental result and analysis are exposed, as well as the SCADA and HMI developed for this particular application. Finally, the conclusion is described in Section 4, which concludes the effectiveness of research method conducted.

2. Research Method

First method that we do in our research is study of the Delineation plant works. In the current system, Figure 3, data was sent or received in plaintext form. The Operator may send the command to the controller with button on the HMI interface. The plant will automate the mixing of two liquid from lower container and upper container. The plant has 4 sensors, S1, S2, S3 and S4 which have a function to detect the level of liquid in the container. The sensor will send the data to the controller, so that the mixing of two liquid can do automatically using response to the actuator, pump and solenoid valve. We also integrate the system with Alarm buzzer to alert for the anomaly system.

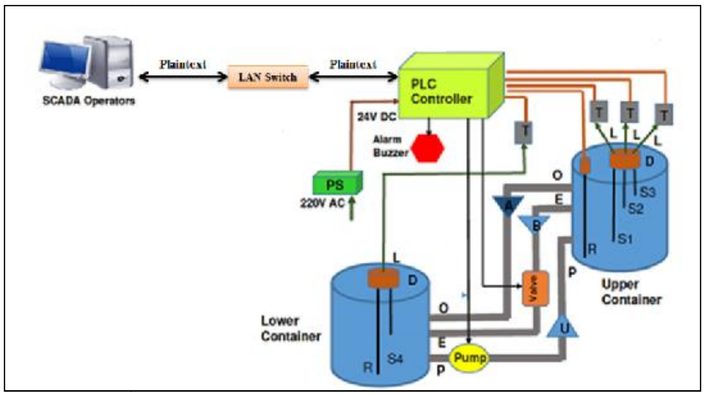


Figure 3. Current SCADA Comunnication

In our current system, it is already provided the HMI (Human Machine Interface), Figure 4, so that Operator can communicate with the controller to start the process. Pump with red color status indicate that the pump is running. Conversely, green color indicate that the pump is stop.

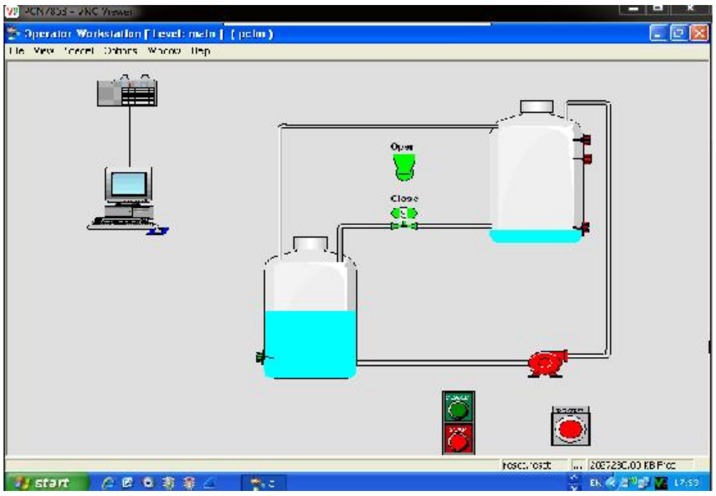


Figure 4. View old HMI

The system that we propose in our experiment is to built an application that can be used to encrypt and decrypt the data communication between SCADA HMI and SCADA server. Because we did not have access to the controller, we implement the ethical hacking methodology [8] to get the data transmitted. In our research, we make a new HMI using Python and Java Programming. Design of new application is secured; we implemented RC-4 algorithm on rapsberry. A new HMI will integrated with Raspberry pi as a gateway for encryption and decryption. The design topologi to be applied are as Figure 5.

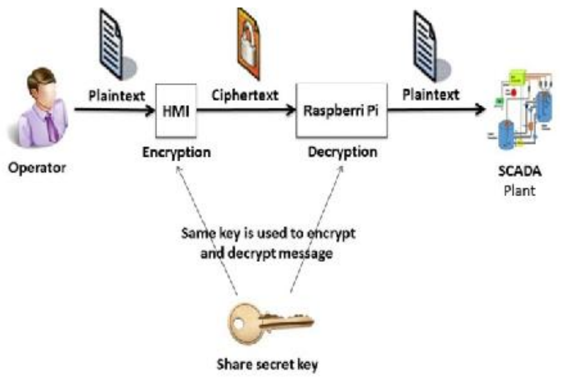


Figure 5. Secured SCADA Communication using RC-4 Algorithm

2.1. Design and Implementation of SCADA Secured Communication using RC-4 Algorithm

Encryption is the process of transforming plaintext data into ciphertext in order to conceal its meaning and so preventing any unauthorized recipient from retrieving the original data. Hence, encryption is mainly used to ensure secrecy. Companies usually encrypt their data before transmission to ensure that the data is secure during transit. The encrypted data is sent over the public network and is decrypted by the intended recipient. Encryption works by running the data (represented as numbers) through a special encryption formula (called a key). Both the sender and the receiver know this key which may be used to encrypt and decrypt the data as shown in Figure 6.

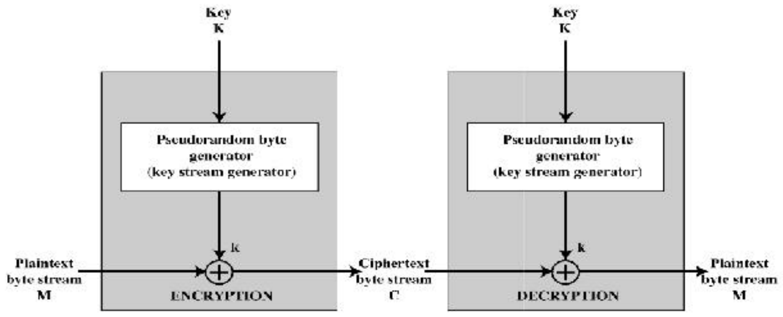


Figure 6. Stream Cipher Diagram

A typical stream cipher encrypts plaintext one byte at a time, although a stream cipher may be designed to operate on one bit at a time or on units larger than a byte at a time. Figure 6 is a representative diagram of stream cipher structure. In this structure a key is input to Pseudorandom byte generator (key stream generator) that produces a stream of 8-bit numbers that are apparently random. A pseudorandom stream is one that is generated by an algorithm but is unpredictable without knowledge of the input key. The output of the generator, called a keystream, is combined one byte at a time with the plaintext stream using the bitwise exclusive-OR (XOR) operation.

The system to be constructed include encryption-decryption program that serves to change the type of message plaintext into ciphertext. The design of the process to be conducted encryption-decryption program described by the flowchart in Figure 7.

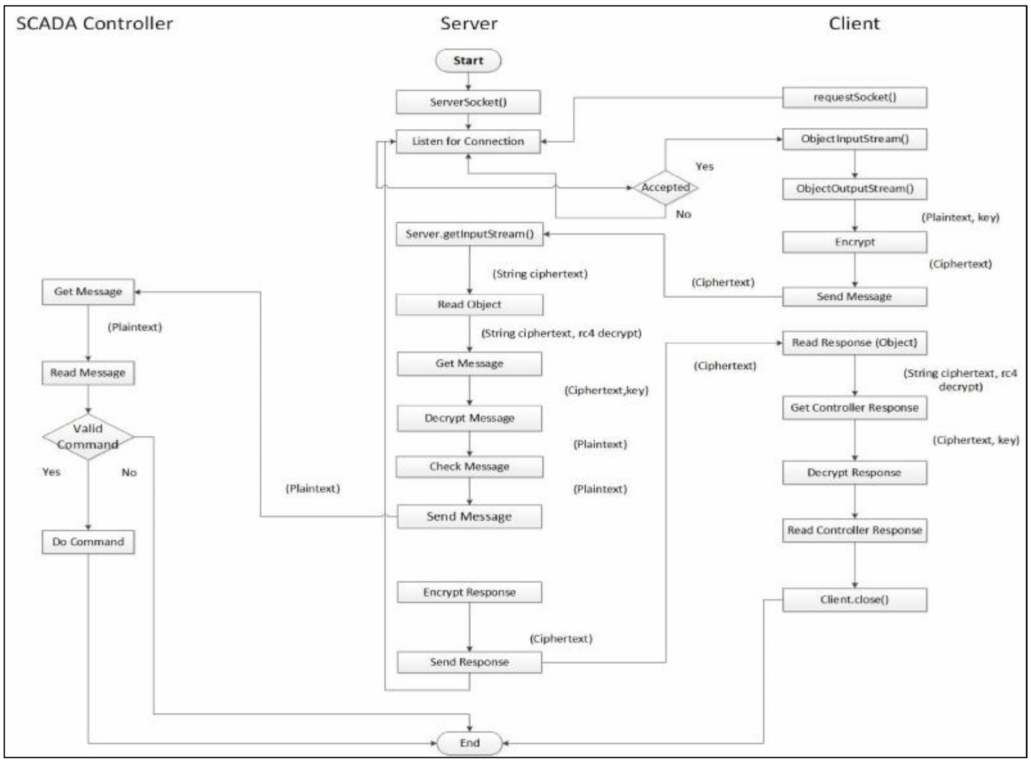


Figure 7. Flowchart Encryption and Decryption

In accordance with the above flowchart, pseudocode is designed to build the system. Pseudocode is created, consisting of programs socket to socket for server and client programs. The following is pseudocode for a server socket program:

```
#Server Socket Program
Declaration:
        serverSocket                    : ServerSocket
        requestSocket, connection: Socket
out,out2                        : ObjectOutputStream
        in,in2                          : ObjectInputStream
message,pesan,key,tangkap: String
        temp                                    : integer
        s[], k[]                        : array of integer
start, end                              : long integer
Class                                           : Server, RC4
Function: main
        run
        sendMessage
        sendRealMessage
encrypt
        decrypt
        initSBox
        setKey
        swap
while true do
        create a server socket on port 2004
wait for a connection from a client
get Input and Output streams
        sent encrypted message to client
        receive and decrypt message from client
                if message equal turn on command then
                send command to plant
                        plant on
                else message equal turn off command then
                send command to plant
                plant off
        send response to client
close socket
endWhile
```

The following is pseudocode for a client socket program:

```
#Client Socket Program
Declaration:
        requestSocket                   : Socket
out,out2                        : ObjectOutputStream
        in,in2                          : ObjectInputStream
message,pesan,key,tangkap,ip,rep : String
        temp                                    : integer
        s[], k[]                        : array of integer
start, end                      : long integer
Class   : Server, RC4
Function:
        main
        run
        sendMessage
        encrypt
        decrypt
        initSBox
        setKey
        swap

while true do
        create a socket connect to server
get Input and Output streams
        receive and decrypt message from server
        send encrypted message to server
        receive and decrypt response from server
close socket
end While
```

3. Results and Analysis

Steps being taken on implementation are in the form of installation, configuration, and creation of program code. The program code is integrated with Java Programming to create a new HMI. Installation and configuration performed on the encryption and decryption of data communication lines with HMI SCADA with Rappsberry Pi as its gateway. Installation and configuration is done with the use of a PC will be carried out with the CentOS operating system.

3.1. Testing Using HMI with Rapsberry Pi as a Gateway

Gateway is a device used to connect one computer to another computer on different networks using different protocols or the same communication between computers so that information can be transmitted. In the experiments conducted, firstly a PC is used as a gateway to connects between the HMI and the Controller. At the gateway, it is implanted a program to perform the encryption and decryption of incoming data. In the experiment we conduct, using PC as a gateway was successfully implemented; the HMI can give a command to controller and get the fast response to the HMI.

The next step should be done is to conduct testing on the design proposed. This testing aims to ensure that the encryption and decryption process with the use of Raspberry Pi as a gateway successfully performed. Here is the new HMI display obtained:

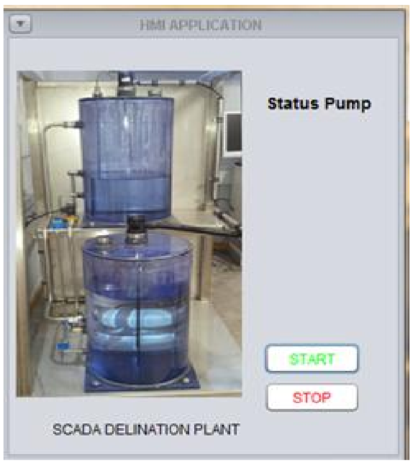


Figure 8. A new HMI using Rapsberry as a gateway

The response time before and after encryption is implemented is show on Figure 9. It can be seen that the average response time before encryption and after encryption is slightly diffrent. We still can control the plant in a good response time.

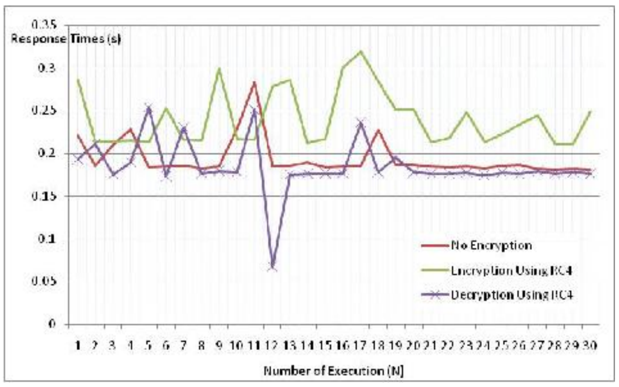
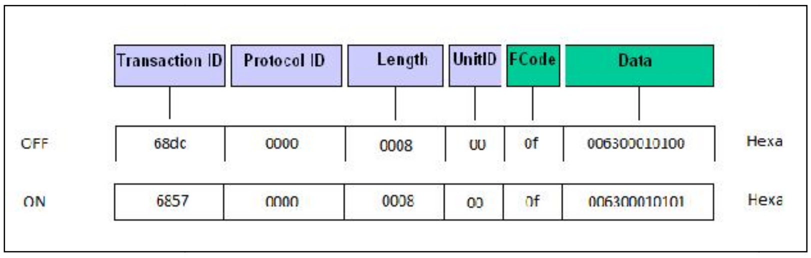


Figure 9. Response Time HMI-Controller

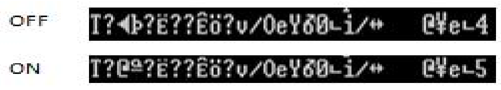
In the new HMI, we provide Start and Stop buttons, so we can give a command to the plant. Start-Stop command has been encrypted, so that communication between the HMI and the controller will be more secure.

Figure 10 below is data transmitted while no encryption is use. Figure 11 is the data transmitted after the algorithm of encryption is applied. From Figure 11, we may see that the data can not be easily interpreted.



	Transaction ID	Protocol ID	Length	UnitID	FCode	Data	
OFF	68dc	0000	0008	00	0f	006300010100	Hexa
ON	6857	0000	0008	00	0f	006300010101	Hexa

Figure 10. On-Off Button: Data Transmitted is plaintext



```
OFF   T?◀Þ?Ë??Êö?ν/0eŸð0ـì/↔   @¥eـ4
ON    T?@º?Ë??Êö?ν/0eŸð0ـì/↔   @¥eـ5
```

Figure 11. Data Transmitted After RC-4 Algorithm Applied

4. Conclusion

The study was conducted on a plant SCADA with Modbus TCP communication. The plant allows the monitoring of the processes that occur for mixing two liquids in Lower Container and Upper Container. In a study produced a communication security of data sent and received is passed through the device Rapsberri Pi by using the RC4 algorithm. So that the transmitted data will be encrypted from the machine operator (master), is passed through a gateway. Rapsberri Gateway will do decryption to the data received, and then the data is sent to the controller. On the experiment, we got a good response time from HMI to Controller. We may conclude that the gateway using Rapsberri Pi can be tested on a larger SCADA plant, so the communication is secure.

References

[1] Eric D Knap. Industrial Network Security: Securing Critical Infrastructure Networks for Smart Grid. *SCADA, and Other Industrial Control Systems*, *Syngress Elsevier*. 2011.

[2] Hahn A, Ashok A, Sridhar S. Cyber-Physical Security Sandboxs: Architecture, Application, and Evaluation for Smart Grid. *IEEE Transaction on Smart Grid*. 2013; 4.

[3] ISA. Security for Industrial Automation and Control Systems Part 1: Terminology, Concepts, and Models. *International Society of Automation (ISA)*. 2007.

[4] J Falco, J Gilsinn K, Stouffer. *IT Security for Industrial Control Systems: Requirements Specification and Performance Testing*. 2004 NDIA Homeland Security Symposium & Exhibition. Crystal City, VA. 2004.

[5] RRR Barbosa, R Sadre, A Pras. *A First Look into SCADA Network Traffic.* In IEEE/IFIP Network Operations and Management Symposium (NOMS 2012). Springer. 2012; 17: 6.

[6] Modbus IDA. Modbus messaging on TCP/IP implementation. Guide v1.0a. 2004.

[7] Gareth Halfacree. Raspberry Pi user guide. Download from http://www.cs.unca.edu/~bruce/Fall14/360/RPiUsersGuide.pdf. 2015.

[8] Albert Sagala, et al. *Study of Hacking Methodology for Gaining Access to SCADA Delineation Plant Network on Modbus TCP Communication*. Under Reviewed on International Conference iceei 2015 – ITB. 2015.

[9] Allam Mousa, et al. Evaluation of the RC4 Algorithm for Data Encryption. *International Journal of COmputer Science & Applications*. 2006; 3(2).