A retrofit network transaction data logger for SCADA control systems

Kalyan Pavurapu

A RETROFIT NETWORK TRANSACTION DATA LOGGER FOR SCADA CONTROL SYSTEMS

By

Kalyan Pavurapu

A Thesis
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in Computer Engineering
in the Department of Electrical and Computer Engineering

Mississippi State, Mississippi

August 2010

A RETROFIT NETWORK TRANSACTION DATA LOGGER FOR SCADA

CONTROL SYSTEMS

By

Kalyan Pavurapu

Approved:

Thomas Morris Assistant Professor in Electrical and Computer Engineering (Director of Thesis)	Robert B. Reese Associate Professor in Electrical and Computer Engineering (Committee Member)
Mahalingam Ramkumar Assistant Professor in Computer & Science Engineering (Committee Member)	James E. Fowler Professor and Director of Graduate Studies
Sarah A. Rajala Dean of Bagley College of Engineering	

Name: Kalyan Pavurapu

Date of Degree: August 7, 2010

Institution: Mississippi State University

Major Field: Computer Engineering

Major Professor: Dr. Thomas Morris

Title of Study: A RETROFIT NETWORK TRANSACTION DATA LOGGER FOR SCADA CONTROL SYSTEMS

Pages in Study: 109

Candidate for Degree of Master of Science

SCADA (Supervisory Control and Data Acquisition) control systems are widely used to control critical processes in various economically and safety critical commercial industries. SCADA control systems are often vulnerable to attacks due to previous industry reliance on security by obscurity to protect control systems. There is a need for an architecture which can log the communications traffic in the SCADA networks. In this work a forensic network traffic data logger retrofit solution for MODBUS and DNP3 network appliances is presented. The data logger uses a bump-in-wire configuration to capture the network transactions, timestamp, cryptographically sign, encrypt and store the network transactions. The data logger is developed to run on embedded and virtual machine platforms. Thus, a retrofit forensic network traffic data logger logs the network traffic in a SCADA control system efficiently without affecting the normal functionality of the control system and the logger data supports post incident forensics analysis.

DEDICATION

I would like to dedicate this thesis to my family and my beloved grandfather.

ACKNOWLEDGEMENTS

I would like to gratefully thank my advisor Dr. Thomas Morris, for his guidance, motivation, constant encouragement and support throughout my research work. I experienced at most pleasure working with SCADA security Lab at Mississippi State. I am thankful to Dr. Robert. B. Reese and Dr. Mahalingam Ramkumar for being part of committee and helped me for successful completion of my master's degree. I would like to thank Southeast Regional Research Initiative (SERRI) and Department of Homeland Security for the financial support during my Master's degree.

Last but not the least, I would like to thank my friends and colleagues at MSU for their help and support. Without their support it would not have been possible to complete this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

SCADA	Supervisory Control and Data Acquisition
MTU	Master Terminal Unit
RTU	Remote Terminal Unit
IED	Intelligent Electronic Device
HMI	Human Machine Interface
PDU	Protocol Data Unit
DNP3	Distributed Network Protocol
HMAC	Hash-based Message Authentication Code
SHA-1	Secure Hash Algorithm
AES	Advanced Encryption Standard
CTR	Counter mode
LAN	Local Area Network
WAN	Wide Area Network
IP	Internet Protocol
CRC	Cyclic Redundancy Check
LRC	Longitudinal Redundancy Check
EPA	Enhanced Performance Architecture

CHAPTER I

INTRODUCTION

SCADA is an acronym for Supervisory Control and Data Acquisition. Modern society has become dependent on critical infrastructure using SCADA such as water treatment and distribution plants, electricity generation and, distribution, petroleum refining and distribution, nuclear energy generation and distribution and manufacturing sector including chemical manufacturing, commercial manufacturing and defense industrial base etc. Failure of these systems can lead to financial and property losses and physical harm to the citizenry. Many of these critical infrastructure use a common set of electronic control systems, collectively called supervisory control and data acquisition (SCADA) systems, to manage complex and potentially dangerous processes. SCADA systems are used to monitor sensors, to open and close valves, and to toggle switches remotely over a networked environment. In typical installations, the network infrastructure itself is poorly protected, leaving the entire SCADA system vulnerable to failure and attacks through malicious activity.

Typical control systems have one or more MTU (Master Terminal Units) which monitor and control downstream nodes called remote terminal units (RTU) and intelligent electronic devices (IED). IEDs are intelligent sensor and control devices which are programmable to meet requirements for a specific process. RTU are more generic programmable devices which allow engineers to customize individual sensor and control connections and correspondingly customize process controls. The master, RTU, and IED nodes are typically networked with MODBUS [1] or DNP3 [2] protocols which communicate over RS-232[3], RS-485[4] serial ports. Additionally, it is common for control systems to be distributed of large distances. In these cases, operators often use wireless communication protocols (often proprietary) or telephony based communications to bridge distances.

SCADA wireless and telephony based communications are often vulnerable to intrusion. Such intrusions are possible because of limited access control capabilities, SCADA operator tendency to use default communication configurations, and the tendency to leave configurations unchanged over long periods of time. Such vulnerabilities have led to at least one documented attack which occurred in Australia in 2000 [5]. In this case, a disgruntled former employee used radio equipment to access and controls multiple sewage plant control system throughout the country. Over a period of three months, the attacker modified control system RTU to create waste water leaks which ultimately led to over 200,000 gallons of raw sewage into local fresh water sources. Because the attacker in this case was accessing the control system RTU had no data logging capabilities over a communication channel unknown to the system operators his accesses went completely undetected and unlogged. If all communications to the penetrated RTU had been logged operators would have easily known an outsider was accessing the system after the initial leaks were detected because logged communication at the RTU would not match log transmissions from known master nodes. Without data logging at the RTU, the actual case, engineers were baffled by the leaks and took 3 years to learn the true source of the problem. Ultimately, the attacker was found after a tip led authorities to search the attacker's home, at which time they found the radio equipment used to carry out attacks. This incident highlights the need for a SCADA data logger which can capture all the SCADA network traffic to prevent attacks on the SCADA system and use this logged information in post forensics analysis. The data logger can also be used to as sensor input for Intrusion Detection System (IDS) solutions and to aid the system debugging.

1.1 SCADA Data Logger Overview

The thesis objective is to design the SCADA data logger which logs the network transactions in the SCADA control system network and the logger data supports the post forensic analysis. The SCADA data logger meets all the requirements listed in the table 1.1.The SCADA Data logger is implemented on both the embedded platform and the virtual machine platform. The embedded platform data logger uses the Xilinx FPGA board and network traffic is logged to a compact flash card available on the FPGA. Whereas, the virtual machine data logger is implemented as Linux process running on the personal computer (PC) and uses either the internal hard disk space or the external memory attached to the PC. In both the cases, the logged data is securely stored using cryptographic methods HMAC [6] (underlying function SHA-1[7]) and Advanced Encryption Standard (AES) [8] with counter mode (CTR) [9] based on the American Gas Association Standards. The SCADA data logger supports SCADA protocols MODBUS and DNP3. The data logger operation is active with the network, but still it delivers the complete traffic unmodified and in time. The data logger is operated in two modes namely fast forward-store mode and store-forward mode. In the store-forward mode, the traffic from master to slave and vice versa are retained in data logger until the other operations such as, time stamping, signing and encrypting the data. Once all the above operations are completed the transaction is sent out from the data logger. Where as in fast forward mode, the data logger receives and transmits byte by byte from one end (Master) to another end (Slave) and vice versa. In this case, the operation such as time stamping, signing and encryption does not affect the normal functioning of the control system. The validation of the SCADA data logger is done by testing the working model at Mississippi State University (MSU) SCADA Security Laboratory. The MSU SCADA Security Laboratory consists of 2 masters PC and 5 slaves connected and controlled by the HMI interface. The model SCADA systems (2 masters and 5 slaves) and the data logger work on all the 5 slaves. Also the values of round trip time latency are measured on three configurations: Loopback, Test PC to embedded platform and Test PC to virtual machine platform data logger. In The test PC to embedded platform data logger and test PC to virtual machine platform data logger, the data logger operates in fast forward-store and Store-forward modes. The loopback configuration is used as baseline for comparison study and value of the roundtrip time latency in this configuration is lowest compared to other configurations. The test PC to embedded platform and test PC to virtual machine platform configurations helps in finding the delay caused by the data logger. Based the comparison studies between the above configurations and modes, the data logger operated in fast forward mode is faster than store-forward mode and has less impact on the overall latency of the system and virtual machine platform is efficient compared to the embedded platform.

1.2 SCADA Data Logger Requirements

Table 1 lists a set of requirements for SCADA data logger. First requirement is to support the bump-in wire configuration. The data logger should be retrofit module that fits in the legacy SCADA systems and does not distributes the normal functioning of the

SCADA system. Second requirement is to support the post forensic analysis. The post forensic analysis helps in finding the reasons for various attacks and intrusions detections etc. As the data logger supports the post forensic analysis, the data logger should protect the network transactions by it. To ensure the protection of the data, the process of time stamping, signing and encrypting of data is done.

Third requirement platform ensures that data logger works on both Personal computer (PC) and embedded platforms with some minor changes in the hardware abstraction layer. The implementation of SCADA data logger helps to run the data logging as a Linux process on the HMI and log the network transaction even on the HMI side. The embedded version data logger helps logging the network transactions at RTU/IED level in remote sites. The selection of the platform is dependent on other parameters such as cost, processing powers and easy deployments etc. Fourth requirement is protocol compatible, the data logger should work independent of the SCADA protocols (MODBUS, DNP3) used for the communications between the master and slave nodes. Fifth, Harmless requirements guarantees that the active operation of the data logger does not affect the normal functionality of the control system. This requirement is proved by validating the SCADA data logger at MSU SCADA Security Laboratory and through measuring the latency values in different configurations. The SCADA data logger does not interfere with the normal functioning of the control system and it adds minimal delay to network transactions.

Table 1.1 SCADA Data Logger Requirements

No	Requirement	Description
1	Bump-in-wire retrofit	The data logger should be able to fit legacy SCADA system in bump -in- wire configuration
2	Support Post forensic analysis	The stored data on the data logger should help in post forensic analysis and thus the protection of the data is highly required, which achieved using signing and encrypting the data.
3	Platform Compatible	The data logger should be compatible to work on both the PC and embedded platforms
4	Protocol compatible	The data logger should be independent of the SCADA protocol used in the SCADA communications i.e Modbus and DNP3
5	Harmless	The security retrofit module should not affect the normal function of the control system
6	Data Acquisition	This requirement ensures that the complete transactions between the master and slave are acquired.
7	Log Data	This helps in time stamping the network transactions while acquiring, signing and encrypting the data.
8	Store Data	The process of storing data involves the selection of the storage media depending on the type of platform used for data logger, size of memory etc
9	Offline analysis	This involves the extraction of data from the stored transactions using cryptographic techniques
10	Support Active analysis	The bump-in wire configuration makes the data logger active with network, which helps in supporting the active analysis in future

Sixth, the data logger must be able to acquire the data from the network efficiently without errors in the data. The SCADA data logger supports data acquisition based on RS232 and RS 485 physical connections. Seventh, log data is to prepare the data acquired to storage. This requirement guarantees properties like time stamping, data integrity and data privacy. The time stamping of the data helps in future analysis of the stored data and data integrity, privacy are achieved using the cryptographic functions such as Hash message authentic code (Underlying function SHA-1) and Advanced Encryption Standard operating in (counter mode).

Eighth requirement, store data is actual process of storing data to memory and uses it for future forensic analysis. It is dependent on the type, lifetime and reliability of the storage media used for the storage. Ninth requirement, the offline analysis is the actual process of the extracting the data from the storage data. As the data logger uses the signing and encryption of data along with timestamp while storing the data in the memory. The offline analysis involves the process of decrypting the stored data, verifying the signature and gives the original transaction for the post forensic analysis. Finally, the data logger should be able to support active analysis. This helps in integrating the efficient intrusion detection system (IDS) to data logger and scanning for the intrusions online by reducing the logging data.

Thus the SCADA data logger satisfies the above requirements and does not affect the normal functioning of the control system The SCADA data logger is implemented on both an embedded platform (FPGA) and the virtual machine platform, which can efficiently log the SCADA communications. The embedded platform data logger can be used at level, where as the virtual machine platform data logger version can be implemented as Linux process running in a virtual machine on the same PC which hosts the HMI. Both version of data loggers helps in logging network transactions the logged information can be used for in post forensic analysis.

1.3 Thesis Organization

Organization of thesis chapters is presented in this section to provide an overview of presented topics in this research work. The second chapter consists of background and details related to SCADA system, SCADA protocols and their related topics. Chapter 3 explains about the basic functional structure of the SCADA data logger, and need of data logging in SCADA systems. Chapter 4 presents the motivation behind the development of SCADA data logger, proposed architecture of data logger and its requirements, implementation details and testing environment. Chapter 5 presents the results based on latency values in various configurations on different platforms. Discussions and analysis were presented on the basis of the results obtained. Chapter 6 concludes the research work and suggests the future on this research topic.

CHAPTER II

OVERVIEW OF SCADA CONTROL SYSTEMS

2.1 Introduction

This chapter gives a brief introduction and discussion about important topics related to thesis subject. SCADA (Supervisory Control and Data acquisition) control systems are widely used to control critical process in various economically and safety critical commercial industries. The major components of SCADA system, evolution of SCADA architecture in various generations and protocols such as MODBUS, DNP3 which plays vital role in the communications between the components of the SCADA system are explained in this chapter. The common attacks on the SCADA system such as response injection attack, command injection attack and denial of service attack are explained. This chapter includes details about the Mississippi State University (MSU) SCADA Security Laboratory.

2.2 General SCADA Architecture

SCADA is an acronym for Supervisory Control and Data Acquisition. SCADA systems are widely used to monitor and control a plant or equipment in industries such as Petro chemical, manufacturing, power generation and distribution Telecommunications, water and waste control, energy, oil and gas refining etc. Figure 2.1 shows the general architecture of the SCADA system. The main components of the SCADA system are the Master Terminal Unit (MTU), Remote Terminal Unit (RTU), Human Machine Interface (HMI), Historian, Communication Channels. Each component has diverse functionality and design. The Control center is the hub of SCADA operations. Control center consists of master terminal units, HMI (Human Machine Interface), Historian and shared resources.

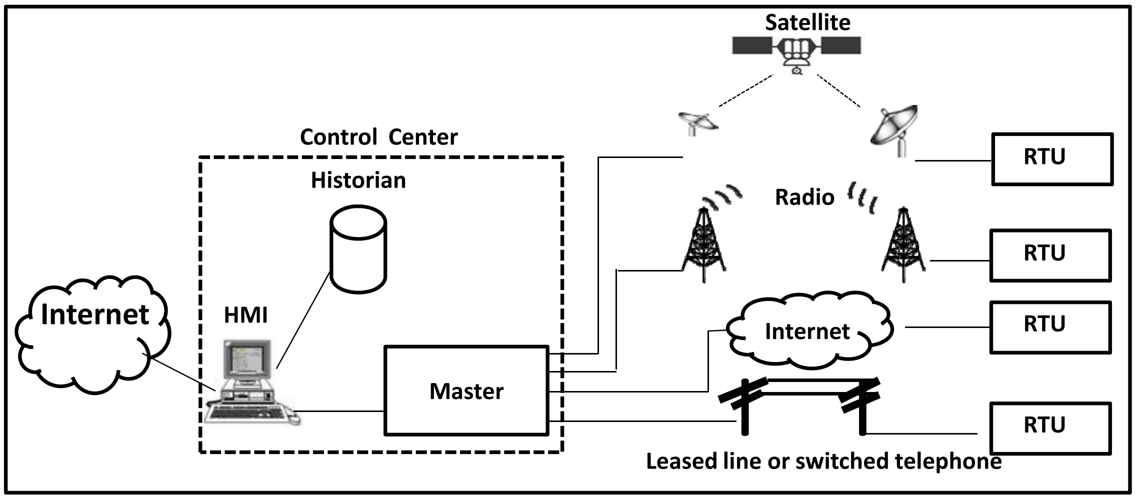


Figure 2.1 General Architecture of SCADA System

Generally, in the SCADA structure, a single master exists which communicates with a hierarchy of sub masters and slaves. The communication between master - slave is initiated by master and slave gives the response. Master and sub master stations have plentiful of memory and processing resources; because they are commonly personal computers or industrial pc's running WINDOWS, LINUX, UNIX etc.

The HMI is the software component that allows the operators to interact with a SCADA system. The functions such as monitoring, controlling, modifying the control process and devices settings. The HMI give a pictorial representation of the SCADA system and displays process status information, reports etc. HMIs for SCADA systems can be accessed using Desktop PCs, web browsers etc. For example, in the Mississippi

State University SCADA Security Laboratory, the HMI software is installed on a desktop PC using it the entire setup can be controlled.

The historian is a database of the historical data from the SCADA system. The historian is updated by the master and the data on the historian can be accessed by the HMI. The stored data from the historian can be retrieved by sending queries from the HMI. Generally, the Historian is located in highly secured places, because it contains all the information about the SCADA system configuration, field devices settings and also some logs entries. If an attacker takes control of this historian, the total functioning of the control system is affected. Thus protection of historian is highly required.

Remote terminal unit (RTU) is a programmable device that interfaces the physical world with the control system (SCADA) and can be customized by the engineers. RTU monitors both the analog and digital parameters, transmits the parameters to central master station. In general, RTU are connected to field devices such as actuators, switches and sensors. The major function of the RTU is to receive the information from the field devices and convert in to language that is compatible to the SCADA system communication protocols (MODBUS, DNP3 etc). Some of features of the RTU are have limited memory, less processing power, and can run only fixed programs(c or ladder logic).

The communication network is established between the master and field based RTUs. The communication network can be wired or wireless networks. Different media like leased lines, dial-up phones, Internet, Radio and satellite are used. In general, leases lines and dial phones are economical to cover a large geographical area. Radio and internet can be used to communicate with remote site devices and is economical compared to leased lines.

Thus the SCADA is the mainly used to control and monitors the various control system and processes. Master terminal Unit, Remote Terminal unit and communication network are the vital components in the SCADA system.

2.3 SCADA System Generations

Initially SCADA system were developed to support basic control system process and later various generations of SCADA system evolved with advanced features. The three generations of the SCADA systems are Monolithic SCADA System, Distributed SCADA System and Networked SCADA System [10].

Monolithic SCADA system is considered as the first generation SCADA systems. In this SCADA system, a single centralized master and several slaves exist. The communications between the master and slaves are implemented using The WAN (wide Area Network). Simple functions such as monitoring and controlling the RTUs are supported in this generation. The main disadvantage in this generation is the hardware software and peripheral are vendor specific only.

The second generation of the SCADA architectures is distributed SCADA system. In this generation, the processing is distributed among the multiple stations. Each station in the system has specific assigned function. These stations are connected with Local Area Networking (LAN).The distribution of the functions among various station increases the processing time, redundancy and reliability of the SCADA system. For example, some station works as communication processors, calculations processors, database base systems, MI and etc. Even if any station fails to work, then another station can be used to control the system without any delays. The communication between the field based RTU and master is still dependent on the WAN and even the components are vendor specific.

Networked SCADA system architecture is open rather than vendor specific and it is the current generation. Open standards and protocols are used, which helped to distribute the SCADA functionality across the WAN. The communication between the master and slave RTUs can use WAN protocols and Internet Protocol (IP).The third party devices like printers, hard drives etc., can be easily connected to the system. Due to usage of standard protocols like internet protocol for communications between master and slave, many SCADA systems are accessible from internet. As a result SCADA systems are more prone to cyber-attacks.

2.4 SCADA Protocols

SCADA system uses standard protocols such as MODBUS and DNP3 for communication between the Master Terminal Units and Remote Terminal Unit (RTU). DNP3 is specially developed for the SCADA system communications. The description about each protocol standards such as addressing rules, message formats etc are explained in the following sections.

2.4.1 MODBUS Protocol

MODBUS protocol is an Application layer protocol [11]. It supports serial and TCP/IP communications. A master-slave model consists of single master and slave. The master issues explicitly commands to one of the slave nodes and process the request. Generally, slave will not transmit data without request from the master and a slave does not communicate with each other.

The MODBUS address space comprises 256 different addresses. The MODBUS master node has no specific address and each slave have unique address on the serial bus. The total address space is divided in to three different parts: Broadcast address (0), Slave address (1 to 247) and Reserved (248-255).The MODBUS address space is shown in figure 2.2.

Broadcast	Slave Address	Reserved
0	1-247	248-255

Figure 2.2 MODBUS Address Space

MODBUS over serial line is a master-slaves protocol. This protocol takes places at second layer of the OSI Model. The general representation of MODBUS serial communication compared to 7-layer OSI model as shown in figure 2.3.

OSI MODEL	**MODBUS Serial Line Protocol**
Application	Modbus Application Protocol
Presentation	EMPTY
Session	EMPTY
Transport	EMPTY
Network	EMPTY
Data link	MODBUS serial line Protocol
Physical	RS232,RS-485

Figure 2.3 MODBUS Protocols and ISO/OSI Model

At the physical level, the communications using different physical interfaces (RS232, RS485).RS232 interfaces can be used for the short distance communications.

RS485 Two –Wire interface is commonly used physical interfaces and RS485 four wire interfaces can be used as extended option.

The MODBUS data link layer consists of two sub layers: Master-slave protocol and Transmission modes. The MODBUS operated in two modes: ASCII and RTU mode.

The MODBUS serial line protocol is master-slaves protocol. In general, only one master is connected to bus and several slaves are connected to same serial bus. The process of communication between the master and slaves is initiated by master. A single MODBUS transaction is initiated by master. The communication between the master and slaves occurs in two modes: Unicast Mode and Broadcast Mode

In the Unicast mode, the master addresses one single slave a same time. The slaves are addressed by their respective address values assigned to them. Each slave has unique address and values range from 1 to 247.In this mode the complete transaction contains of two messages: a request from the master and response from the slave.

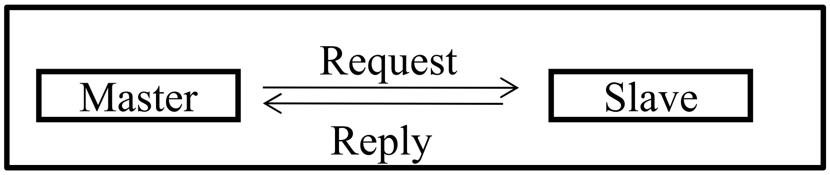


Figure 2.4 Unicast Mode

In the broadcast mode, the master addresses many slaves at same time. The address 0 is reserved for the broadcast exchange between the master and slaves. The MODBUS transaction in this mode consists of single message: a request from the master and no replies from the slaves. Generally this mode is used for writing commands to slaves.

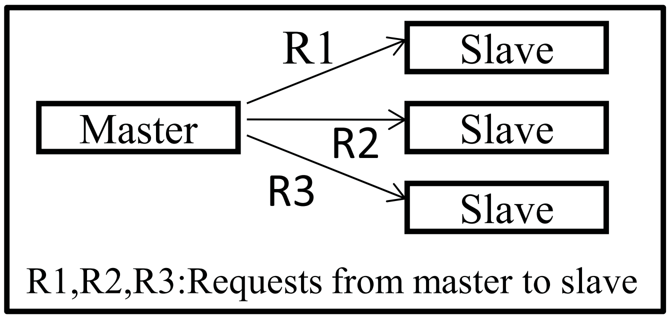


Figure 2.5 Broadcast Mode

Serial transmission modes defines the bit contents of message fields transmitted serially and also explains the process of packing the data in to message fields. The transmission mode and serial port parameters must be same for all the devices on a MODBUS serial line. MODBUS serial transmission can be operated in two modes: RTU Transmission mode and ASCII Transmission mode.

In this RTU Transmission mode, each message is transmitted in a continuous stream of characters. The 8 bit binary coding system is followed in RTU mode. The bit sequence of the RTU Mode with parity and no parity are shown in the figures (2.6) and (2.7).

Start Bit	Data Bits								Parity bit		Stop Bit
	1(LSB)	2	3	4	5	6	7	8(MSB)	Even	Odd	

Figure 2.6 The RTU Bit Sequence(Parity Bit)

From the figure, the start bit indicates the start of the message. The data bits consist of actual data to be transmitted. The data is transmitted from LSB (Least Significant Bit) to MSB (Most significant bit) i.e., from left to right. The total number of data bits in frame is 7.Parity bit is used for parity checking. Even, odd and no parity can be used depending on the configuration of the system. The default parity checking mode is even parity checking. The stop bit indicates the end of the message. In general if there is no-parity, an additional stop bit is added to the bit sequence.

Start Bit	Data Bits								Stop Bit	Stop Bit
	1(LSB)	2	3	4	5	6	7	8(MSB)		

Figure 2.7 The RTU Bit Sequence(No Parity Bit)

The maximum size of the RTU Message frame is 256 bytes. Each RTU frame is transmitted with a start and stop point. The interval between each frame is 3.5 char, which helps the devices to identify the start and stop of the message. If the interval is less than or equal to 1.5char, the message is incomplete and discarded by the receiver. The entire message is transmitted as a continuous stream of characters. The RTU message frame consists of the following fields: slave address, function code, data and CRC.

Slave address is 1 byte in length and gives details about to which slave the message is addressed. Data field contains the actual data of message. Cyclic redundancy check method field is sub divided into two fields: CRC (Low) and CRC (High).The value of CRC is independent of parity checking method applied and does not include stop, start bits in its calculation. The CRC is calculated at both transmission and receiving devices.

The ASCII mode is used in the communications when the RTU mode cannot be implemented. In this mode, each 8 bit byte is sent as two ASCII characters. The bit sequence of the ASCII mode is shown in the figure 2.9.

Start >=3.5 char	Slave Address	Function Code	Data	CRC		Stop >=3.5 char
	1 byte	1 byte	0-252bytes	2 bytes		
				CRC (low)	CRC (High)	

Figure 2.8 The RTU Message Format

Start Bit	Data Bits							Parity bit		Stop Bit
	1(LSB)	2	3	4	5	6	7(MSB)	Even	Odd	

Figure 2.9 ASCII Bit Sequence(Parity Bit)

The Start bit in the bit sequence indicates the start of message. The data bits consist of actual data to be transmitted. The data is transmitted from LSB (Least Significant Bit) to MSB (Most significant bit) i.e., from left to right. The number of data bits is 7. This bit is used for parity checking. Even, odd and no parity can be used depending on the configuration of the system. The default parity checking mode is even parity checking. The stop bit indicates the end of the message. An additional bit is added to the bit sequence if no-parity checking mode is applied.

Start Bit	Data Bits							Stop Bit	Stop Bit
	1(LSB)	2	3	4	5	6	7(MSB)		

Figure 2.10 ASCII Bit Sequence(No Parity Bit)

The maximum size of the ASCII message is 513 characters. Each RTU frame is transmitted with a start and stop point. In the ASCII mode, a message should start with colon (:) character (ASCII 3A HEX) and the end with carriage return and line feed (CRLF pair)(0D0A).The devices monitor continuously for the colon. when the colon is received, the device decodes the next characters until the end of the frame is obtained. In the LRC field in the ASCII message holds the calculated LRC values. The LRC value is calculated by the transmit device and appended at the end of the message. The device that receives recalculates the LRC and cross checks it with the actual LRC value. LRC calculation does not include the start and stop bits.

Start	Address	Function code	Data	LRC	Stop bit
1 char	2 chars	2 chars	0-2*252chars	2 chars	1 char

Figure 2.11 ASCII Message Format

MODBUS function codes are the elements of the MODBUS request/reply PDUs. MODBUS function codes are divided in to three categories: public function codes, User-defined function codes and reserved function codes. Public function codes are unique, well documented, publicly documented and validated by the MODBUS-IDA community. There are three ranges of the public function codes, i.e.1-65, and 72-100 and from 110-127 decimal. User-defined Function Codes are user defined and not supported by the specification. There are two ranges of the user-defined function codes, i.e.65 to 72 and from 100 to 110 decimal. Reserved Function Codes are not part of the specification and range from 128-255. The format of reserved function code is function code/sub function code or just function code. All the sub function codes ranging from 0-255 are reserved.

The differences between the both MODBUS transmission modes: MODBUS ASCII and MODBUS RTU are listed in the table 2.1.

Table 2.1 Differences Between ASCII and RTU Modes

ASCII	RTU
Messages are in readable format	Messages are in binary coding and cannot be read.
It allows a time gap of 1 second between the bytes of message.	Continuous streams of messages are sent
The start and end of the message are identified with special characters like semi colon(:), and CR/LF combination	A minimum gap of 3.5 char is present to indicate the start of each message.
The bandwidth utilization in this mode is twice that of RTU mode.	The bandwidth utilization is less than the ASCII mode

Thus, MODBUS is the SCADA protocol used for serial communication and can be operated in two modes: MODBUS ASCII and MODBUS RTU.

2.4.2 DNP3 Protocol

Distributed network Protocol (DNP3) is an open standard protocol [13]. It was developed by Harris, Distributed Automation Products. This Protocol is widely accepted communication protocol in SCADA systems. These involve acquiring the information and sending information between physically separate devices. The DNP3 protocol is used to transmit relatively small packets of data in reliable manner compared to general purpose protocols such as FTP, which can send large files.DNP3 is widely used in oil & gas, waste water and security industries. The DNP3 offers a wide range features such as time stamping of the message, time synchronization, supports peer to peer topology, multiple master topology, secure file transfers and many more[14].

DNP3 uses the enhanced performance architecture model developed by the IEC technical committee [15].The EPA model uses two hardware layer, two software layers and application layer. DNP3 uses the three layers of the EPA model and adds some extra transport functions. The layers in the DNP3 are the following:

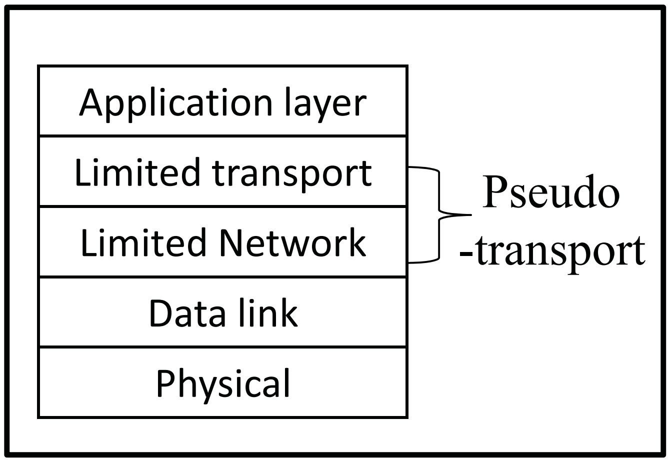


Figure 2.12 DNP3 Enhanced Performance Architecture Model

The physical layer defines the physical interface characteristics in terms of electrical specifications, timing, pin-outs etc. The standard recommended physical layer standards for DNP3 are 8 data bits, 1 start bit, 1 stop bit, no parity and Rs232C voltage levels and control signals. The data link layer provides for the reliable transmission of data across the physical medium. The main features of the DNP3 data link layer are frame synchronization, multiplexing, data fragmentation, error Checking and also supports multiple masters and peer to peer communicationsDNP3 have a Pseudo-transport layer which supports the functions of both network and transport layer. The network layer provides the functions like flow control and routing of the network packets. The transport layer supports the function like message assembling and disassembling functions, error correction etc.

The DNP3 application layer has following features such as secure File Transfer, Date and Time Synchronization and also supports the polling based on the data priority. The DNP3 is particularly developed for SCADA systems. It is designed to optimize the data acquisition and control commands transmission between the devices. The DNP3 have the following advantages such as high data integrity, minimized Overhead, robust, efficient, secure and self-Compatible.

Thus the MODBUS and DNP3 are mostly commonly used industry SCADA protocols in the SCADA systems. The SCADA data logger supports above two protocols: MODBUS in ASCII mode, MODBUS in RTU mode and DNP3.

2.5 Attacks on SCADA systems

SCADA systems are first developed to meet the basic requirements of the control systems, where security has less importance. However, the increase in the demand for connectivity between a SCADA and other networks components like internet and corporate network exposed the critical parts of SCADA to the public. Threats can come from numerous sources such as malicious intruders, disgruntled employees, equipment failures etc. Some of the potential attacks against SCADA system are such as, response injection, command injection and denial of service Data and control injection [16].

First attack is response injection, this kind of attack involves injecting false responses into a control system. In the control systems, the processes are dependent on the feedback control loop which monitors physical process data before making control decisions, protecting the integrity of the sensor measurements from the physical process is critical. So injection of the false responses makes the control algorithms and operators or dispatchers to take a false decision. In this attack, the attacker impersonates as slave and sends fake responses to the master and forces it to take false decisions. Second attack is command injection attack in which false controls commands are injected in to control system. In this kind of attack, the attackers impersonates himself/herself as master to slaves and makes an attempt to issue false command control to the control the operations of slaves. Generally, the remote terminals and intelligent electronic devices are programmed to monitor automatically and control the physical process directly at the remote sites. The ladder logic and c language are available programming language to program the remote terminals and intelligent electronic devices. Even the attacker can make an attempt to issue control commands to change or overwrite the ladder logic and c programs running on the remote terminal devices.

Third type of attack is denial of service is a kind of attack in which the making the system unresponsive and unavailable to the other systems. In the SCADA system, both the master and slaves can be attacked using this attack. In the both cases, the one component becomes unresponsive and unavailable to other component. One example of denial of service are flooding the system with continuous requests and making the system unavailable for other illegitimate requests. An attack is documented which occurred in Bellingham in 1991[17]. In this case, 250, 000 gallons of gasoline is leaked in to streams and later created explosion, because at the time of incident the total control system became unresponsive and not able to control the pressure relief value. The reasons for the failure is not known due unavailability of the data logging schemes in the control system

In all the above mentioned attacks the attacker first sniffs the data transmitted over the network and gains some confidential information from the network. This is possible in the SCADA network because the SCADA network communications are not unencrypted. Using this data, the attacker establishes the connection between the master and slave and acts as man in middle. The configuration for the attack is shown in the figure 2.13.

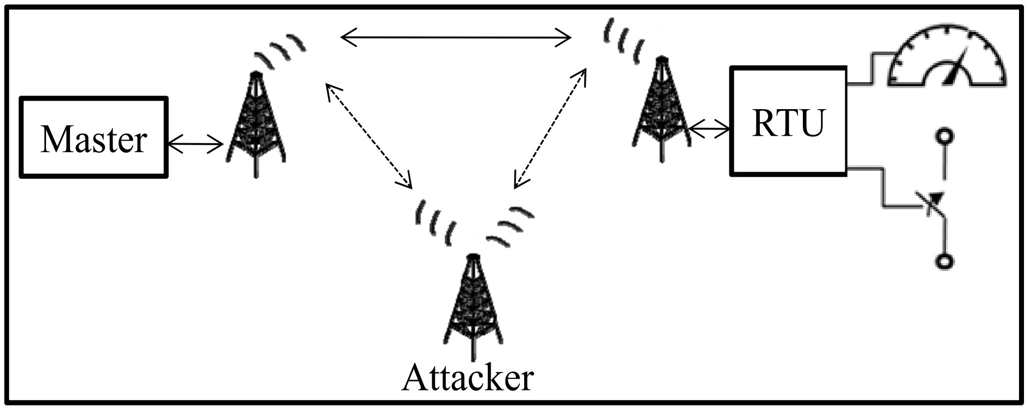


Figure 2.13 Communication with and without attacker

Figure 2.13 shows the normal communications between the master and slave in the absence of attacker and communication with attacker as man in middle. As man in middle, the attackers impersonates himself/herself as master to the slave and as a slave to the master. Now the attacker can send false data to master faking the readings and responses from the slave (Response Injection Attack) and even he can make the changes in the control message from master to slave which may cause malfunctioning of the devices (Command Injection Attack). The malfunctioning of the devices and false readings from the slaves might ultimately effect the over functioning of the control system. Even the attacker can launch denial of service attacks and make the one system unavailable/unresponsive to other systems. Thus a solid security strategy should be developed to prevent these attacks on SCADA system and safeguard the national property.

2.6 Mississippi State University (MSU) SCADA Security laboratory

The Mississippi State University (MSU) SCADA Security Laboratory was used to validate the data logger. The MSU SCADA security laboratory consists of two masters and five slaves. This laboratory was designed and constructed using actual commercial control system components representative of those found in general use within the critical infrastructure. The MSU SCADA Security laboratory itself is configured to model a gas pipeline operation, an industrial blower (such as found in mining applications), a conveyor belt operation, a municipal water tower, and an oil/gas storage facility as slaves and two HMI interfaces to control these slaves. The master and slaves stations use MODBUS and DNP3 protocols to communicate with each other. Both the master and slaves can be operated in the two modes of MODBUS i.e., ASCII and RTU modes. The transmission medium used in the laboratory is 900 MHz free wave radios[18], RS232 and Ethernet. The software configuration of the laboratory contains of Allen-bradley RSLogix 500 Programming software [19], Telepace [20], SCADAPack Programming software [21] and iFIX PC monitoring and control software [22]. The SCADA main station and other slaves connected to it have SCADAPack lightPLC, Freewave 900 MHZ radios and requires a power supply of 24VDC.

The overview of the MSU SCADA Security Laboratory including the description about the master and slaves configuration is explained below. Figure 2.14 shows the HMI interface of complete laboratory setup at the master end. The two masters are used to control the slaves connected to them. Both the masters communicate with the slaves using freewave 900MHZ radios.

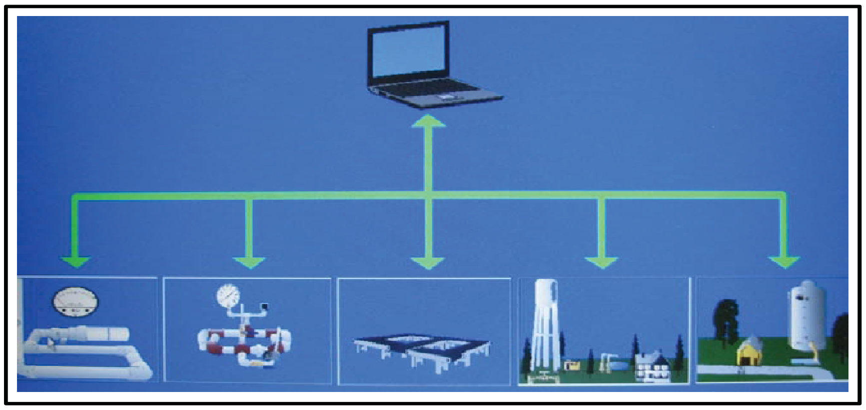


Figure 2.14 The Human Machine Interface

The iFIX PC monitoring and control software is installed on the Laboratory PC to see the pictorial representation of complete system. Using iFIX PC monitoring and control software, the operator at master end can control the operation of all slaves. The HMI software is capable of monitoring and controlling all slaves simultaneously and individually. Various parameters of the slaves can be controlled by the HMI interface. If the communications is Lost between the HMI and slaves, the slave RTU continues to function with the last set of parameters provided by the HMI. When the control system is operated based on the MODBUS protocol, the lab supports both the unicast mode and broadcast mode. In the unicast mode, the master communicates with only single at a time, whereas in the broad cast mode the master sends out request commands to all the slaves connected it at same time.

The slaves connected to the two masters are gas pipeline operation, an industrial blower, a conveyor belt operation, a municipal water tower, and an oil/gas storage facility.

First, the gas pipeline system is model of the industrial pipeline transportation used for transporting the gas and liquids. Figure 2.15 (a) shows the gas pipeline control system model the lab and (b) shows the representation of the gas pipeline control system on HMI interface The mock gas pipeline control system consists of a gas pipeline, a pump, a sensor to read the pressure values, and an actuator to control the pressure relief value. This gas pipeline slave communicates with master using free wave 900 MHz radios. The pressure level in the pipeline is maintained between the maximum and minimum parameters values which RTU holds The RTU continually measures the pressure and sends updates of pressure values to the master at regular intervals.

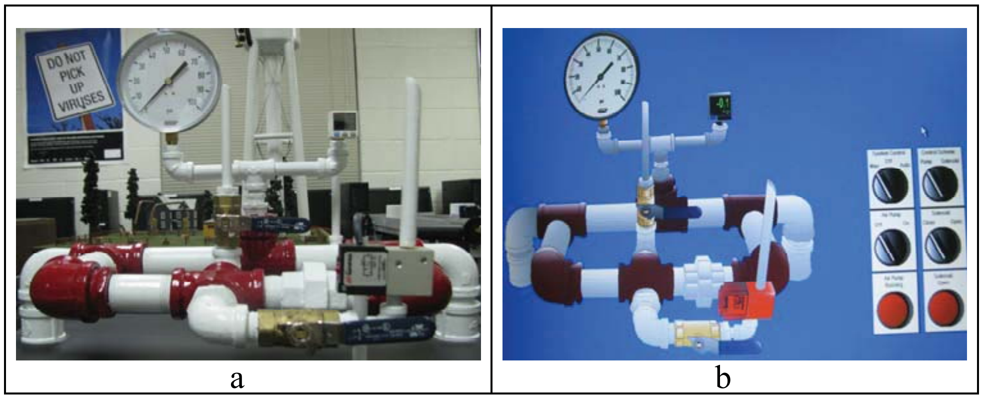


Figure 2.15 The Gas Pipeline Control System

The slave can be operated in two modes: manual and automatic mode. In the automatic mode, the on/off of motor is controlled automatically depending on the pressure values. For example, if the pressure reaches the threshold value, the master issues a command to turn off the motor. In manual mode, the motor is controlled by the operator depending on the pressure value. Using the HMI interface, the various operations such as open and close the pressure relief value, set maximum and minimum parameters turn off and on the entire system.

Second, the industrial blower is the model of the blowers used in the mining applications. Figure 2.16 (a) shows the industrial blower control system model the lab and (b) shows the representation of the industrial blower control system on HMI interface. The mock blower system consists of air evacuation tube, a pump, air flow sensor and an actuator to open an air duct. The RTU controls the air conduct and pump to maintain air flow.

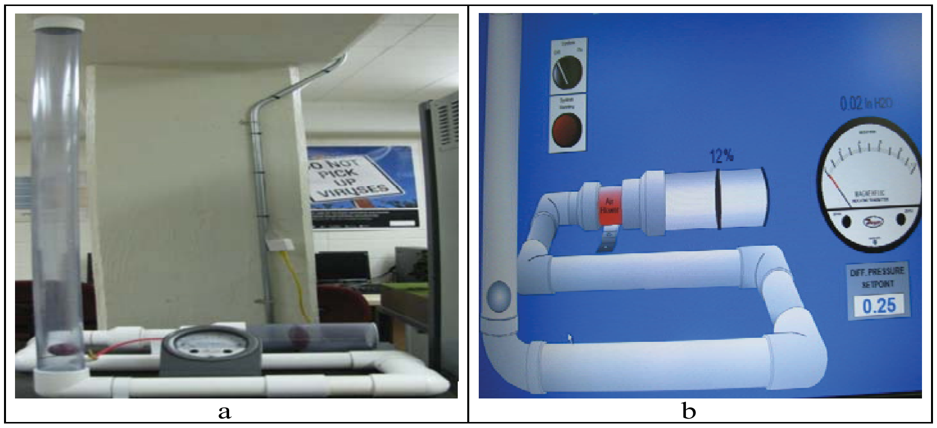
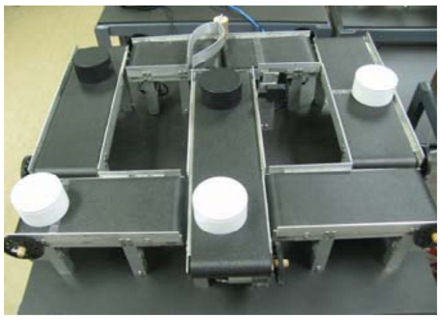


Figure 2.16 The Industrial Blower Control System

The pressure value in the tube is continually reported by the salve to master at regular intervals. The HMI monitors and displays airflow of the system on the operator's PC screen. The blower can be operated in both manual and automatic mode.

Third, the conveyer belt model is similar to industrial conveyer model used in many industries for material handling, transporting various industrial and agricultural products. Figure 2.17 (a) shows the conveyer belt control system model the lab and (b) shows the representation of the conveyer belt control system on HMI interface. The conveyer belt system in the lab consists of three conveyer belts, diverter, photo eye and pucks (black and white) and used to sort the pucks by color.


a

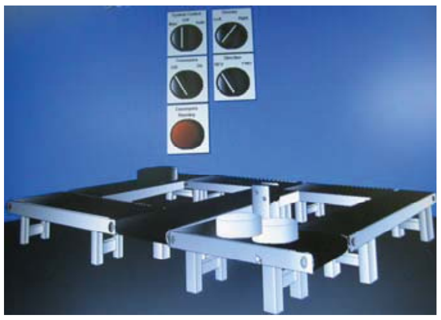
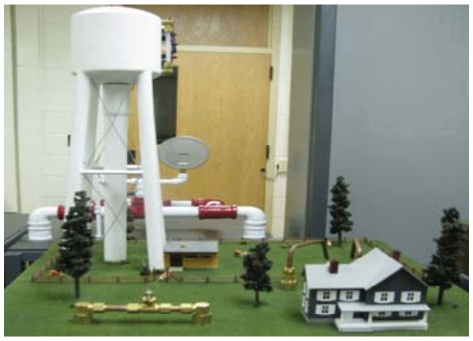
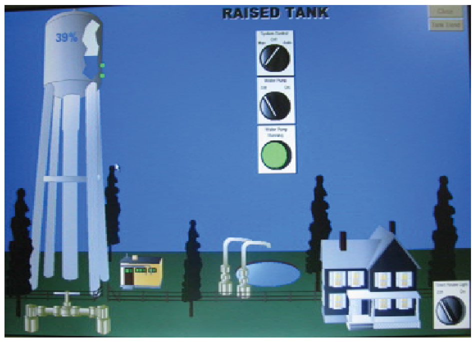

b

Figure 2.17 The Conveyer Belt Control System

The conveyer belts can be operated in both clock wise and anti clock wise directions. The photo eye is used to detect the color of puff and diverter is used to separate the puffs based on their color. The HMI monitors the sensor values and uses conveyor belt speed to display the approximate location of the pucks on the conveyor belt. This system can also be operated in automatic mode and manual mode in which the operator takes control the system and controls the sorting operation.

Fourth, the municipal water tower models a water tower used in water distribution systems.


a



b

Figure 2.18 The Municipal Water Tower Control System

Figure 2.18 (a) shows the municipal water tower control system model the lab and (b) shows the representation of the municipal water tower control system on HMI interface. The mock water tower in the lab consists of the elevated water tank, water pump, air flow sensor, and water level sensors mounted at four different levels on the elevated water tank. The sensors inside the water tank updates about the level of water in the tank The RTU makes an attempt to maintain water between the 2 middle water sensor levels by turning on and off the water pump. If the water reaches either the highest sensor level or lower sensor level then alarms are triggered on the HMI. Depending upon the level of water in the tank, the process of turning water pump on/off is controlled in both the automatic and manual modes.

The last model in the lab is oil/gas storage unit. This model represents actual industrial oil/gas storage tanks. Figure 2.19 (a) shows the oil/gas storage control system model in the MSU SCADA Security lab and (b) shows the representation of the municipal water tower control system on HMI interface.

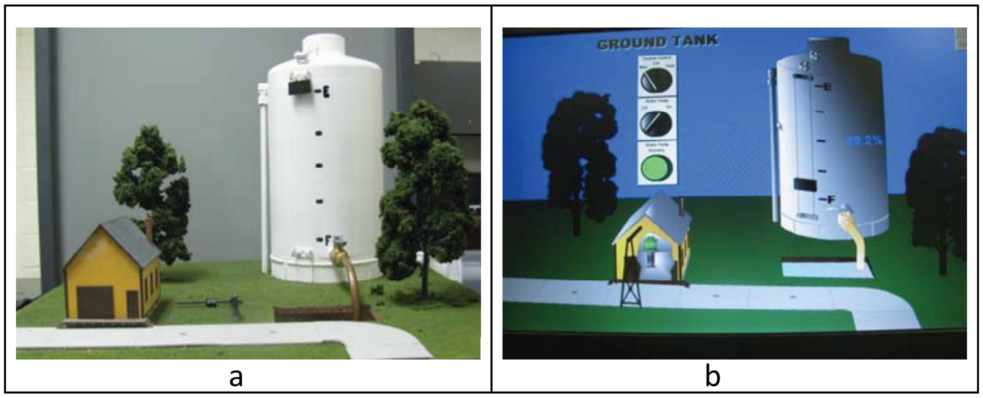


Figure 2.19 The Oil/Gas Storage Control systems

The mock oil storage system consists of oil storage tank, a pump and sensors monitoring the levels of oil/gas in the tank. The RTU contains HI and LO oil level parameters and attempts to maintain the oil level between those limits. The HMI interface consists if HI/HI and LO/LO level parameters and alarms are triggered if the oil level reaches the HI/HI or LO/LO levels. This system can also be operated in both automatic and manual modes.

Thus the MSU SCADA security laboratory is model of SCADA system similar to the industry and was used to validate the working of the SCADA data logger.

2.7 Summary

This chapter introduced all the basics and background information related to SCADA system, protocols used in the communication between the master and slaves, and various attacks on the SCADA system. Details about a MSU SCADA Security laboratory are also introduced in this chapter.

CHAPTER III

DATA LOGGING IN SCADA NETWORKS

3.1 Introduction

This chapter explains details about general data logging, the need of data logging in the SCADA networks and functional structure of the SCADA data logger. Need for integrity and confidentiality requirements in the implementation of the data logger and ways to achieve the requirements are explained in this chapter.

Data logging is the measurement and recording of parameters over a period of time such as network traffic, temperature, humidity, light, pressure, voltage, power etc [23]. The main idea behind the data logging concept is to store the information for future references and analysis. For example, the black box in the airplane is a data logging device. It collects all the parameters related to plane performance. This collected information can be used later performance analysis and details about failures of devices.

SCADA control systems may have highly involved master nodes in which the master continually polls RTU and IED for sensor data and directly controls the process or control may be distributed to downstream RTU/IED which control the process. Control systems are typically networked control system specific communication protocols such as MODBUS and DNP3. MODBUS and DNP3 are carried on various physical layers including RS-232, RS-485, Ethernet, and many proprietary radio solutions. These systems are more prone to attacks at several points in the network such at RTU level, HMI etc. Many documented attacks cannot provide the exact evidence for the cause of the attacks. So a data logging scheme is required for the SCADA system which can log the network transactions and use this data in the later post forensic analysis of various incidents. SCADA control system data loggers should be able to monitor and log communications traffic in the network to allow an intrusion detection tool to analyze logged network traffic and detect unauthorized penetrations. The major function of the SCADA data logger is not only limited to the logging the data and storing it, but also to protect the logged data to support the post forensic analysis. The requirements of SCADA data logger such as data integrity and data confidentiality ensure the data stored by the data logger is secure. To meet the above requirements, HMAC with SHA1 as underlying function is use to provide the data integrity and data confidentiality is achieved using the encryption function (AES operating Counter Mode). The HMAC SHA1 and AES –CTR functions are implemented by the SCADA data logger and safeguard the logged data for later forensic analysis.

3.2 Data logging in SCADA Network

The SCADA control data loggers should monitor and log all communication traffic to and fro from the master (MTU) and slave (). Unauthorized penetration of SCADA control systems may happen at many points. Various attacks on SCADA systems are explained in section 2.6. For example, RTU and IED communication network can be penetrate. It is common for control systems to use radio communications to connect master to RTU over large distances. Often these radio connections are unencrypted connections with no access control. Intruders with matching radio equipment may access control systems via these radio links and take control of the RTU/IED and may lead to devastating consequences on the control system operation. An exploit of this type has been documented in [24] and involves response injection attack and denial of service attacks. In this exploit, attacker uses a radio and connects as a rogue slave to master. The rogue slave is able to communicate with master and can monitor the communications between the master and other slaves. By using the denial of service attack, the network is flooded with continuous data. As a result, the other slaves are unable to communicate with MTU at this moment. During this period, if a critical situations such increase in pressure value, overflow of the oil etc occurs at the slave end, then the MTU cannot control the situation as the communication is lost and results in huge damages. Therefore it is necessary to log network transactions both at upstream and downstream of the SCADA network actions. Typical RTU and IED do not have storage capability to log their actions and they also commonly do not have processor resources available to dedicate to the logging function. Generally, the upstream control system has high processing computers, workstations and a huge database known as historian. The historian does not store the information related to the network transactions, but it stores information about login failures, business documents etc. Hence to overcome the various kinds of attacks, the SCADA data logger is implemented on two platforms: virtual machine platform and embedded system platform.

In [25] Chandia et al. a forensic architecture for the SCADA network is proposed which captures the network traffic at various locations. The exact location of various forensic agents is not addressed. Our architecture aims at placing the data loggers in different locations to complete capture network transactions in the SCADA system.

Figure 3.1 shows the architecture of the SCADA control system with SCADA data logger retrofits. This placement of data loggers showed in the figure will capture all the network transactions along with traffic associated with response injection, command injection and denial of service attacks mentioned in the section 2.5.The SCADA data loggers are placed in two locations. First, a virtual machine data logger is running on the HMI in the figure will capture the network traffic associated with the response injection attacks and denial of service attacks on MTU. The response injections and denial of service attacks may originate from an attacker who penetrated the communication link between the MTU and RTU. An attack of this type on a wireless communications is demonstrated in [24].

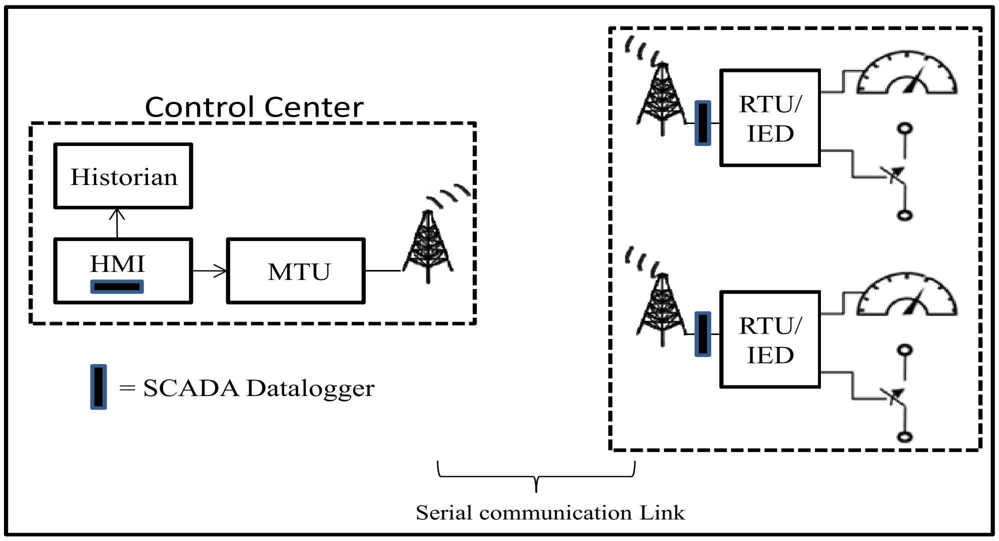


Figure 3.1 SCADA Architecture With Retrofit Data loggers

Second, the embedded platform data logger is placed in the downstream and is attached to the RTU/IED. The data logger is connected in the bump-in–wire configuration. This placement of data logger helps in capturing the network traffic associated with command injection and denial of service attack on RTU/IED. The command injection attacks may originate from a penetration of the corporate network via internet or from an insider and from a attacker who penetrated in to the communication link between the MTU and RTU/IED. The data logger attached to the RTU/IED will capture the network transaction related to command injection attack and denial of service on RTU/IED. This placement of data logger helps in logging all the transactions, such as if the system is configured to communicate with MODBUS protocol, it supports both the broadcast mode and unicast mode. In this unicast mode, the master communicates with a single slave at a time and includes both request from the MTU and response from RTU/IED. The data loggers placed at two points is able to log the both response and request. Similarly, in the broadcast mode the MTU communicates with all the RTU/IED connected it and includes only the request from the MTU. Even in this case, the data loggers are able to log the transactions successfully. Thus, the SCADA data logger implemented on the virtual machine platform data logger running as the Linux process on the HMI host and embedded platform data logger connected to the downstream elements can efficiently capture the network transactions between the MTU and RTU/IED and stores the transactions for future post forensic analysis.

3.3 Functional Structure of SCADA Data Logger

Figure 3.2 shows the general functional structure of the Data logger. In general, the major blocks of the data logging system are the data acquisition, logging and storage, offline analysis, online analysis and display [26].Whereas, the major blocks of the SCADA data logger are data acquisition, logging and storage and offline analysis. At present, online analysis and display functions of general data logger are not supported by the SCADA data logger. This functional structure is common for both the embedded system platform and virtual machine platform data loggers.

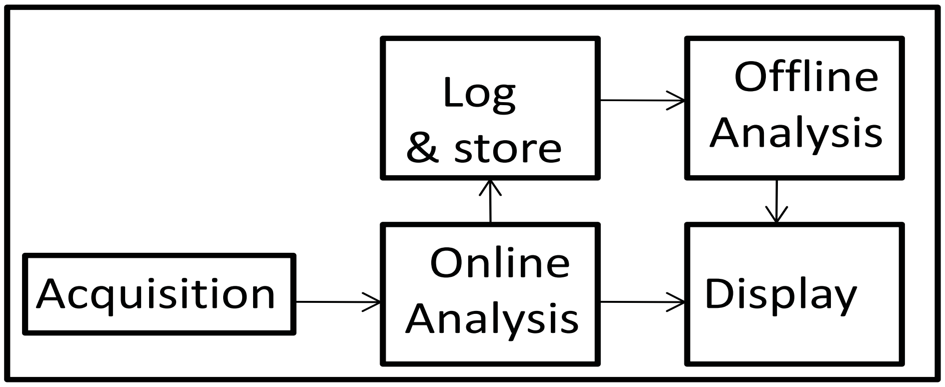


Figure 3.2 General Functional Structure of SCADA Data Logger

The functions of each functional block of SCADA data logger is explained below. First, functional block of the SCADA data logger is data acquisition. The data acquisition is a process of acquiring data from the physical world and sending data to the system to process it. The data acquisition in the SCADA data logger is achieved using the UARTs. UART(s) will collect the traffic between the control system nodes and RTU. Many physical layer connections are available to establish the connection between the master nodes and downstream nodes such as ethernet, RS-232/RS-485.Our data logger design supports RS-232 and RS-485 physical layer connections.

Second functional block of the SCADA data logger is logging and storage. The logging and storage blocks are essential in every data-logging system. Some of the common storage media used is floppy disks, tape drives, network drives, RAID drives and etc. Software plays a vital role in determining the efficiency of the data logging systems, because well developed software determines the storage format, efficient utilization of the memory space on the disk and other factors like writing time, reading time. In general the efficiency of the data logger is dependent on the storage format. The common storage formats available are ASCII text files, binary files and data bases. ASCII text files are the common, flexible and compatible format for storage. These files do not utilize the disk space efficiently and used only for slow data acquisition systems. Binary files are the efficient method of data storage. In this format the raw bytes are stored are stored which takes less space and less processor overhead. This format can be used in fast data acquisition models. Data bases are the most ideal storage format, because of their organized and structured format. Additional features such as data backup, fast data retrieval and inserting etc are supported by the databases.

In SCADA data logger, the logging and storing software in the SCADA data logger is developed by using c language. On the embedded platform, the logging and store software uses the memory card (1 GB compact flash card) for storing the data and later the data can be retrieved from the card. Whereas in the case of virtual machine platform, the transactions are stored on the drive space available on the PC.As the logged data by the data logger should be protected securely to support the post forensic analysis. The software also implements cryptographic algorithms to sign and encrypt the data. The HMAC SHA 1 is used to sign the data and verifies the data integrity and authenticity of the data and the AES operating in counter mode ensures the data confidentiality. The SCADA data logger also supports the time stamping of the data for temporal analysis in offline analysis and to support the offline analysis in future. The storage format supported by the SCADA data logger is ASCII text file and in future the storage format can be extended to support the binary and databases. The software opens each log file for every transaction and also maintains the track of log files opened in one master file.

Third functional structural block of the SCADA data logger is offline analysis. Offline analysis is process of performing analysis on the data that is acquired and stored. This process involves the extraction of data from the logged data using several functions such as mathematical functions, cryptographic functions etc. Some of the offline analysis types are applying basic statistics functions on the data, identifying the false data, classification of data etc. In SCADA data logger, the process of offline analysis involves the extraction of original data from the acquired and stored data. In the SCADA data logger, data retrieval is carried out in different ways such as master directly polls the data logger to access the data, read directly from the flash card and logon as user on the PC based data logger to gain access to the data. After the data is retrieved from the storage, several operations such as decrypting the message, recalculating the MAC value, comparing the MAC values (stored and calculated) are performed on the data.

The next functional component of a typical data logging system is online analysis. In general, the online analysis is accomplished using commercial software and requires high resources. Alarming, event management, intrusion detection are some of the functions of online analysis. At present the online analysis is not supported by the SCADA data logger. Online analysis can be implemented on the data logger as a future work. The online analysis helps in actively analyzing and capturing the transactions. The online analysis reduces the number of packets which must be stored in the data logger memory and helps in efficient utilization of the memory available.

The last function of the data logging is to support display. The main aim of the display function is to view the data acquired by the data logger. Generally, data is classified in to two types: live data and historical data. Live data is data that is viewed while it is acquired by the data logger whereas the historical data is data that is previously acquired by the data logger and stored in the memory. The display function has very less importance in the SCADA network, because generally the RTU and field devices are located in the remote areas where human access is mostly restricted. Thus viewing the live data is not possible at those locations. However, display of the historical data is supported by the SCADA data logger. Thus, the functional structure of the SCADA data logger and functions associated with each block is explained.

3.4 Importance of Data Integrity and Data Confidentiality

The SCADA data logger uses cryptography techniques to achieve the data integrity and data confidentiality requirements of the SCADA data logger. As mentioned before, the SCADA data logger used the concept of signing (HMACSHA1) and encrypting (AES-CTR) of data are used by the SCADA data logger to protect the data and overcome different various attacks. Some of the possible attacks on the storage devices (Flash cards and hard drives) are data manipulation, data insertion etc. The data logger acquires the network transactions from the network, timestamps the data, signs, encrypts the data and finally stores the data. The logged data is used for post forensic analysis. Among various available techniques SCADA data logger uses the techniques such as HMAC with SHA 1 as underlying function to sign the data and AES in counter mode is used to encrypt the data. The stored transaction on the storage device is encrypted data consists of the original message, nonce and message authentication code (MAC). The input for the HMAC-SHA1 is the concatenated data of original transaction, timestamp and nonce. The MAC value is output values from the HMAC-SHA1 function As the transactions is encrypted using AES operating counter mode ensures the data confidentiality and HMAC-SHA1 guarantees the property of data integrity and data authenticity.

The main advantage of adding cryptographic techniques such signing and encryption is to protect the data on the storage devices securely. First, let consider the data logging without applying any cryptographic features to protect the data. In this scenario, the data logger acquires data, timestamps the data and finally stores data on the storage devices. So the storage device consists of only raw network transactions with just timestamp value appended to it. If an attacker gains access over the storage devices, he/her can perform several attacks such as manipulating the existing transactions, adding the new transactions, deleting some field values in the transactions etc. Later when the post forensic analysis is performed, there are no chances of finding the fake data resembling the original transactions. Thus, to overcome the various attacks on the storage devices and support the post forensic analysis, the data logger implementation considers data integrity and data confidentiality as important requirements.

Next, let us consider the data logging which involves cryptographic features such as signing and encryption. In this case, the data logger acquires the network transactions, timestamps the data, signs, encrypts the data and finally stores the data on the storage devices. So the data stored on the storage device is in encrypted form. If an attacker gains access over the storage devices, there is no chance of data manipulation or data insertion etc as the data is in encrypted form. By any chance if the attacker is able to manipulate the data or insert the new data on the storage device, then the signing helps in identifying the original data and fake data. Because, when data is retrieved back from the storage device, first operation of decryption is performed on the data and next the MAC value is recalculated. The comparison between the computed MAC after the retrieval and stored MAC helps to distinguish between the original transactions and fake transactions. First, if the MAC computed tails with the stored MAC value, then it ensures that data is not manipulated and original transaction is safe. Next, if the MAC computed after the retrieval from the storage device differs from the stored MAC, then the transactions is considered as fake data and is discarded. Thus, the requirements data integrity and data confidentiality plays vital role in the designing of the SCADA data logger and protecting the data securely.

3.5 Summary

This chapter presents an overview of data logging and various cryptographic techniques used in the implementation of the SCADA data logger. The importance of the data logging in SCADA networks and the functional structure of SCADA data logger are explained in this chapter.

CHAPTER IV

SCADA DATA LOGGER ARCHITECTURE

4.1 Introduction

This chapter explains the motivation behind the development of the SCADA data logger, related works and proposed SCADA Data logger architecture. Both architecture on PC and embedded platforms are explained with block diagrams. The major requirements of the data logger and ways of achieving them, details about the software architecture and components on the embedded platform are also explained in this chapter.

4.2 Motivation

Supervisory Control and Data Acquisition (SCADA) is a common set of electronic control systems used to manage complex and potentially dangerous process. Typically SCADA control system has MTU (Master Terminal Unit) which monitors and controls the downstream nodes called Remote Terminal Unit (RTU) and intelligent electronic devices (IED). The maser, RTU and IED nodes re networked with MODBUS and DNP3 protocols which over leased lines, dial up phones, radio and internet. SCADA wireless and telephony based communications are often vulnerable to intrusion. Such intrusions are possible because of limited access control capabilities, SCADA operator tendency to use default communication configurations, and the tendency to leave configurations unchanged over long periods of time. Such vulnerabilities have led to at least one documented attack which occurred in Australia in 2000 [5]. In this case, a disgruntled former employee used radio equipment to access and controls multiple sewage plant control system throughout the country. The attacker modified control system RTU to create waste water leaks which ultimately led to over 200,000 gallons of raw sewage into local rivers leaking into local fresh water sources. Because the attacker in this case was accessing the control system RTU with no data logging capabilities over a communication channel unknown to the system operators his accesses went completely undetected and unlogged. If all communications to the penetrate RTU had been logged operators would have easily known an outsider was accessing the system after the initial leaks were detected because logged communication at the RTU would not match log transmissions from known master nodes. Without data logging at the RTU, the actual case, engineers were baffled by the leaks and took 3 years to learn the true source of the problem. Ultimately, the attacker was found after a tip led authorities to search the attacker home, at which time they found the radio equipment used to carry out attacks. Hence there is a need of a SCADA data logger which can captures all the traffic to prevent attacks on the SCADA system and use this logged information for later forensics analysis.

4.3 Related Works

In [25] Chandia et al. propose a forensic architecture which can be used to capture SCADA control system communications for subsequent forensic analysis. In the Chandia architecture, agents capture data at three levels and forward collected traffic to a data warehouse for storage and future analysis. Level 1 agents collect communications to and from the control system master nodes. Level 2 agents collect traffic at intermediate locations in the network. Finally, level 3 agents collect communication traffic from downstream nodes such as RTU and IED. The level 1, 2, and 3 agents capture network traffic and create synopses of the network packets according to a set of predefined configuration rules. Each synopsis contains a time stamp and location details required for the forensic analysis. The level agents forward the synopsis packets to the data warehouse which is located in the upstream network. The data warehouse analyzes each synopsis packet and creates a data signature, which is stored along with the synopsis. The date warehouse supports queries on the stored data. All the communications between the data warehouse and level agents occur on an isolated side channel network. The proposed architecture with level agents and data warehouse is not able to characterize the position of level agent3 in downstream. Chandia et al. do not specify the exact location, intended behavior, or architecture of their level 3 agents. The level 3 agent is most similar to our data logger architecture in that it collects communication at downstream nodes.

Snort is a rule based open source network intrusion detection tool [27]. Snort collects and logs network traffic, analyzes network traffic searching for rule violations, and alerts the administrator of suspicious activity. Snort is commonly used to monitor Ethernet and TCP/IP communications traffic. Rule sets have also been developed to allow snort to monitor and analyze MODBUS and DNP3 traffic between master nodes and RTU/IED. Such implementations can be used on the master nodes where sufficient processing resources are available to run snort. RTU and IED do not typically have the processing power or the storage capabilities to support Snort.

At present, in market some control system vendors are manufacturing the RTU supporting the data logging feature. But, this RTU developed by the vendors supports user to log the data from the RTU to external store device. They does not supports feature of logging the network transactions and use in later post forensic analysis. For example, Control Micro system, Inc [28] offers two SCADA RTU devices with data logging functionality, but they does not log the network transactions in the SCADA systems.

4.4 SCADA Data logger Requirements

The SCADA data logger is implemented to capture all traffic at RTU level and is places at downstream. It is implemented on both the platforms: the PC platform and embedded platform (FPGA).Various SCADA protocols such as MODBUS ASCII, MODBUS RTU and DNP3 are supported by the Data logger. Many requirements are defined for the SCADA data logger and successfully achieved them through implementation, validation of data logger.

The requirements of the SCADA data logger operating on both PC and embedded platform are shown in the Table 4.1. A data logging system must acquire data, log and store the data, provide a mechanism for active or online analysis, provide a mechanism for offline analysis, include mechanisms to ensure data integrity, be continuously available to log data at the rate transactions occur on the monitored medium, compatible to a1l platforms (Virtual machine and Embedded) and protect the privacy of the stored data etc.The first requirement of the SCADA data logger is support the bump-in-wire configuration. Many SCADA systems operating today are deployed decades ago with a major concern to control the critical processes in the control systems. The security of these systems is not considered at the time of deployment. Thus these systems are vulnerable to a variety of attacks, leading to huge damage due to the failures of critical infrastructures.

The lifetime of the various SCADA equipments spans several years from the installations (20 to 50 years) and replacing this equipment with new ones incur huge costs and is not economically infeasible. Hence, the economic solution is to connect the security retrofit devices in bump-in wire configuration to these legacy systems and achieve the required features such as security etc. The SCADA data logger is a security retrofit module connected in the bump-in-wire configuration. As the data logger is connected in the bump-in wire configuration, it makes the data logger to be active with network. Being active with network helps the data logger to support the active analysis and integrate the intrusion detection system (IDS) with the SCADA data logger.

The second requirement of the SCADA data logger is support the post forensic analysis. The SCADA systems are vulnerable to a variety of attacks and reasons for the attacks are unknown from the post forensic analysis. For example in the case of two documented attacks, one in Australia in 2001 [5] and another incident in Bellingham[17].The investigation to find the reasons of attack took long time and the exact reasons for the attacks are unknown due to lack of proper data logging facility. Thus, the SCADA data logger supports the post forensic analysis as it logs all the communication traffic between the master and slave. To support the post forensic analysis, the data logger should protect the data logged safely and securely. To safeguard the data, the data logger acquires the data, timestamps the data, both the operations of signing and encryption are performed on the data and finally stores the data. The time stamping of data helps in finding the time of incident occurrence and the signing of data, encrypting data ensures the cryptographic properties such as data integrity and data privacy respectively.

Table 4.1 SCADA Data Logger Requirements

	Requirement	
1	Bump-in-wire retrofit	The data logger should be able to fit legacy SCADA system in bump –in- wire configuration
2	Support Post forensic analysis	The stored data on the data logger should help in post forensic analysis and thus the protection of the data is highly required, which achieved using signing and encrypting the data.
3	Platform Compatible	The data logger should be compatible to work on both the PC and embedded platforms
4	Protocol Independent	The data logger should be independent of the SCADA protocol used in the SCADA communications i.e Modbus and DNP3
5	Harmless	The security retrofit module should not affect the normal function of the control system
6	Data Acquisition	This requirement ensures that the complete transactions between the master and slave are acquired.
7	Log Data	This helps in time stamping the network transactions while acquiring, signing and encrypting the data.
8	Store Data	The process of storing data involves the selection of the storage media depending on the type of platform used for data logger, size of memory etc
9	Offline analysis	This involves the extraction of data from the stored transactions using cryptographic techniques
10	Support Active analysis	The bump-in wire configuration makes the data logger active with network, which helps in supporting the active analysis in future

The third requirement of the SCADA data logger is it should be compatible to work on the PC in virtual environment and embedded platforms. The SCADA data logger implemented works on the virtual machine platform and embedded platform. The same software code works for the both platforms except a few changes in the hardware abstraction layer. First, the virtual machine version data logger is used to log the network transactions in the upstream component such as HMI. This version is implemented in such way that it can run as a Linux process on the same PC which hosts HMI. The virtual machine platform data logger can efficiently log the network transactions related to response injection and denial of service attacks on MTU.

Second, the embedded platform data logger is connected to the downstream components (RTU/IED). Generally, the RTU/IED has very little processing power, memory storage and is not able to support to the logging facility. So this embedded platform can be added as the retrofit to the downstream components to log the network transactions. The SCADA data logger is connected to the RTU/IED in a bump-in-wire configuration and adds the logging facility at the downstream (RTU/IED) level. This placement of the data logger at level can efficiently logs the network transactions related to command injection and denial of service attacks on s. Thus the SCADA data logger is compatible with virtual machine platform and embedded platform.

The fourth requirement is SCADA data logger is to independent of the communication protocol used. In SCADA system, the master, RTU and IED communicate with standard protocols such as MODBUS and DNP3.All the components should be configured to communicate over a single protocol. For example, a master configured to work on MODBUS protocol issues MODBUS commands and only the slave which operates on same protocol responses back to the commands from master. An external device connected to the SCADA system should be able to support these protocols. The SCADA data logger supports both the protocols MODBUS and DNP3.It also supports the MODBUS RTU mode and MODBUS ASCII mode operation. Thus, the SCADA mode is protocol independent and supports all the SCADA protocols (MODBUS and DNP3).

The fifth requirement of the SCADA Data logger is being harmless. The data logger should not impede the functionality of the control system process being monitored. The SCADA data logger is active with network. As a result the communications between the master and slave passes through the data logger and cause little delay. But still the overall control system functioning is not affected by adding the data logger in bump in wire configuration. Generally, in the SCADA systems, the master starts communication with a command and slave responds to the command. The master waits for some specified amount time depending on the type of the control system. For example, in electric sub stations, the master polls for every 2-4 seconds and in our MSU SCADA Security Laboratory the master polls for every 1 sec. If the response does not arrive within the time limit, the master issues a new command to slave. Thus any retrofit that fits in the bump-in-wire configuration between the master and slave adds latency, but it should be able to deliver the commands and response within time limit.

The effect of the SCADA data logger on the control system in terms of the latency delay is explained below. The figure explains the difference in communication with and without SCADA data logger. From the figure, let t1 be the time at with the master send the command (cmd1) to slave and t2 is the time at which the response (Res1) reaches the masters from the slave. The difference between the master and slave gives the round trip latency (t2-t1) and the master issues another command after it gets the response. This process continues between the master and slave at regular intervals. By adding the SCADA data logger to the SCADA network, it introduces minimal delay without any interference to the normal communications between the master and slave. Let t3 be the time at which a command (CMD2) is used to slave and t4 is time when the response (RES2) is received without data logger. Let td be the time delay added by the SCADA data logger. As a result the master receives the message after a delay of td. Even adding the data logger does not affect the communication between the master and slave because it adds minimal delay of 40 ms and hence the response reaches the master in time before the next command issued.

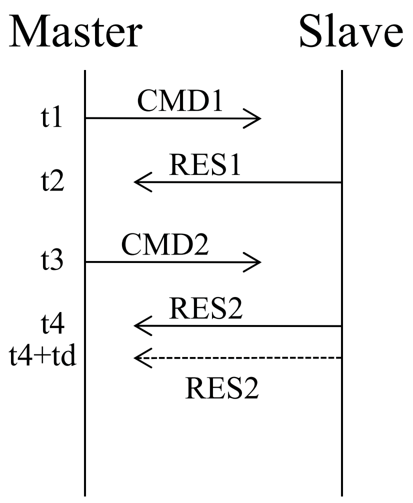


Figure 4.1 The Effect of SCADA Data Logger in terms of Latency

Even though the SCADA data logger is active with the control system, it does not modify any data in the messages corresponding to the control system transactions and delivers complete transaction to the SCADA components (MTU, RTU). It just logs the transactions and stores to the memory. The bump in wire configuration of the SCADA data logger and normal configuration of SCADA system is shown in the figure 4.2.

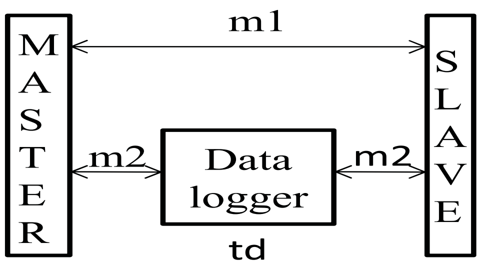


Figure 4.2 Difference Between Normal Communications and Communications with Data Logger

The above figure shows two configurations: one the normal communications between the master and slave and other is with data logger in bump-in-wire configuration. Both the configurations are tested in the SCADA security lab at Mississippi state university and proper functioning of the control system is observed in both cases. Let The m1 is the message transactions between the master and slave without data logger to log the transactions. M2 is the message from master to data logger in bump-in-wire configuration. Td is the time delay added by the data logger to communications. In the normal configuration case, as there is no device to obstruct the communication the control system functions normally and has no delay. In bump-in wire configuration, the data logger is positioned between the master and slave. First the data is received from the master (m2) and logs the data, later forwards it to slave (m2).Still in this configuration, the message (m2) is not modified by the data logger sent from the master. It just adds a time delay (ts in milliseconds) and delivers completely to slave. The SCADA data logger is harmless and does not affect the normal functioning of the control system.

The sixth requirement of the SCADA data logger is data acquisition. The data logger acquires two types of data. First, the data logger uses serial UART(s) to collect RS-232/RS-485 traffic between control system master nodes and the RTU or IED. Second, the data logger samples and stores field device data by directly monitoring field device signals. Third, the data logger can generate MODBUS read transactions to poll information from the RTU/IED for storage. RS-232/RS-485 is a common physical layer connection between master nodes and downstream. However, other physical layer types are also in use. For example, many systems are beginning to use Ethernet as a physical layer connection. Our current data logger design supports RS-232 and RS-485physical layer connections.

Seventh requirement of the SCADA data logger is to log data. Logging is the act of preparing acquired data by adding security features before storing. The logging act have three sub requirements such as data integrity, data confidentiality and time stamping. First sub requirement of the logging is time stamping. When the data is acquired and time stamped which helps to support the temporal analysis in both the active and offline analysis. Second sub requirement of the data logger is to guarantee the integrity of data. Generally Hashing functions are used to provide the data integrity. The SCADA data logger used HMAC-SHA1 to ensure the data integrity for the logged data and support post forensics analysis. Figure 4.3.a shows the flow diagram of the steps followed in the data logging process while storing the network transactions and figure 4.3.b shows the process involved while retrieving the data for the post forensic analysis. When the original message(Tm) is received, the message is appended with timestamp value (tTM) and nonce value (n). The concatenated message is hashed with the HMAC-SHA-1 and the output MAC value is appended to it. During the offline analysis, the data is retrieved from the stored transactions and value of the MAC is recalculated and compared with stored MAC. If both the MAC values are equal, then the data is not modified in the storage media and it ensures the data is protected safely. In case, if the MAC value recalculated and compared with stored MAC value and are not equal, then it indicates some data on the storage media has been modified. So in this way it is easy to protect the data on the storage media and enhance the security feature of the SCADA data logger.

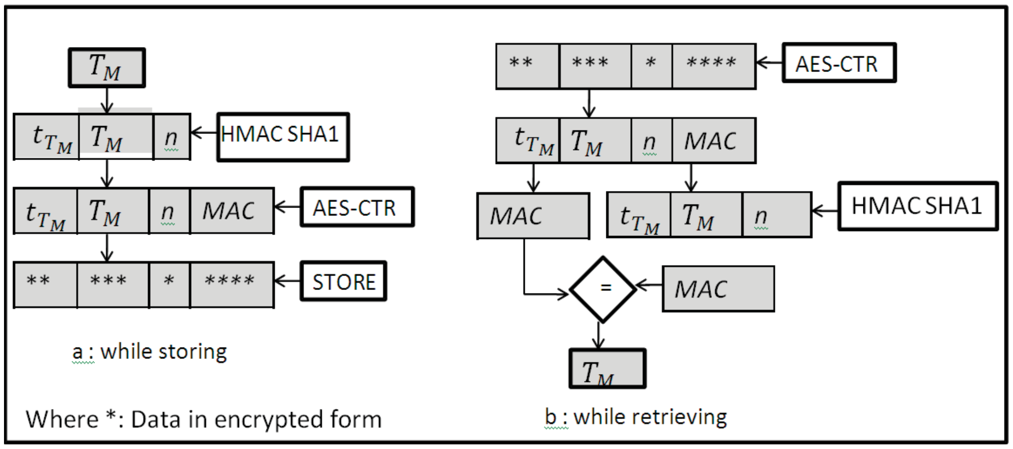


Figure 4.3 Flowchart For Data Storage and Data Retrieval

The last sub requirement of the logging is data confidentiality. Encryption of the message guarantees the confidentiality of the message and the SCADA data logger uses the Advanced Encryption Standard (AES) operating in Counter mode with 128 bit key. The concatenated result of original message (T_M), time stamp value (t_{T_M}), nonce and the MAC from output of the HMACSHA1 algorithm is encrypted using AES_CTR_128. Thus three sub requirements of the logging ensure the message with proper time stamping, and provides data integrity and confidentiality.

$$T_l = e_{k_2}(\{T_M, t_{T_M}, H_{k_1}(\{T_M, t_{T_M}, n\}), n\}) \qquad 1$$

Equation 1 shows a logged transaction (T_l) after preparation for storage. The original MODBUS transaction (T_M) is appended with a time stamp (t_{T_M}) and a random nonce (n). The concatenated result is hashed with an HMAC [6] function (computing with an underlying SHA-1 function) using key (k_1). Next, the captured MODBUS transaction (T_M) is concatenated with the hash result and the nonce. The concatenated result is encrypted using AES-CTR and result is stored on the hard disk drive. Where k_2 is the encryption key. The HMAC, the nonce are added to ensure data integrity. The time stamp appended to the logged transaction is added to support temporal analysis in both active and offline analysis. The system master should periodically synchronize the real time clock in the data logger with real time clocks in other nodes in the control system to support correlation of data logger results from different points in the system. The frequency of time synchronization depends upon the drift of the various clocks in the system. The AES-CTR-128 achieves the data confidentiality to the logged information by the data logger. Thus logging and its sub requirements are important in preparing acquired data for storage and enhance the security features of the data logger.

The Eighth requirement of SCADA data logger is to storage data. Storage as the act of storing acquired data and its accompanying log information for later retrieval. This storage is dependent on the performance, reliability and size of the storage media used. Current technologies offer two alternatives for the file storage: hard disk drives and. flash memory. Hard disk drives are a complicated dynamic system that consists of the electronic, electrical and mechanical components [29]. Hard drives provide large capacity at low cost and through put is high for large files. The main disadvantages of the hard disk drives are high energy consumption, high read latency. Flash memory is the non volatile integrated circuit which can hold the data and is electrically erasable [30]. Flash memory has advantages such as low latency, low energy consumption, high throughput for read access, light weight, portable etc. Thus flash-based storage devices have tremendous potential to replace the hard disk drives. The SCADA data logger works on both platforms virtual machine and embedded platforms. In the case of the embedded platform, the data logger uses the compact flash card available on the FPGA and the virtual machine platform data logger uses the hard drive disk to store the transactions.

The SCADA data logger works for all the SCADA communication protocols (MODBUS ASCII, MODBUS RTU and DNP3). Equation 1 shows the logged transaction (T_l) which includes original transaction plus 72 byte of log information. The timestamp is 8 bytes which provides room to support timestamps of 585K years, when stored with microsecond precision. The HMAC-SHA1produces a MAC value of 32 bytes and nonce of 32 bytes used which matches the key length. The size of the logged transaction and number of the transactions logged is dependent on the type of the protocol used. In general in the case of electrical substations, the SCADA master polls the devices connected to it for every 2 to 4 seconds. At each 2-4 second there will be a command from the master top slave requesting the data and a response providing data from slave to master. Let us see the various protocol logged transactions size and number of transactions logged per a fixed memory size. The possible transactions are MODBUS ASCII, MODBUS RTU and DNP3. First, in the case of MODBUS ASCII mode the maximum size of the transaction is 513 bytes and combining them with the log data the total size of the logged transaction will be 585. The data logger operating in the MODBUS ASCII can store 1.8×10^6 transactions per gigabyte. For the 2 second polling case, the data logger can store 2.8 years of MODBUS ASCII transactions per gigabyte and for 4 second polling case the data logger can store for 5.6 years. Second, in the case of MODBUS RTU mode, the maximum size of the transactions is 256 bytes and adding the log data to the original transaction the total size of the transaction is 328 bytes. The data logger operating in the MODBUS RTU can store 3.27×10^6 transactions per gigabyte. In the 2 second polling case, the data logger can hold data for a period of 5.0 years and for 4 second polling case it can hold for a period of 10 years. Final, the in the case of DNP3 protocol supporting data logger, the maximum size of the original DNP3 transactions is 292 bytes. The maximum size of the logged transactions is 364 bytes which includes the 292 bytes of original transactions and 72 bytes of log information. In this case, the data logger can log 2.95×10^6 transactions per gigabyte and for 2 second polling case the data logger can hold the data for a period of 4.5 years. For 4 second polling case, the data logger can log data for a period of 9 years.

In the case of MSU SCADA Security Laboratory, the Master polls every slave connected to it for every one sec. First, in the case of the MODBUS ASCII, the data logger can log network transactions for a period of 1.4 years per gigabyte. Second, the data logger operating in MODBUS RTU Mode can log the traffic for a period of 2.5 years. Finally, the data logger operating on the DNP3 protocol can log the network transactions for a period of 2.25 years.

For the above analysis, the compact flash card is feasible for the embedded platform data logger and whereas the data logger running on the HMI host may monitor the network transaction between the HMI and many downstream s. So it may require more storage capacity to log all the network transactions. Since this version of data logger runs on the HMI host PC, it can access the system hard disk space to store the data and providing significantly more storage capacity. The SCADA data logger supporting all the SCADA communications protocols satisfies the NERC CIP data logging storage standards. The NERC CIP 005 -3requires the data logs to be maintained for 90 calendar days [31]. In least polling scenario, the three protocols (MODBUS ASCII, MODBUS RTU and DNP3) transactions with log information can hold data for 1.4,2.5 and 2.25 years respectively. Thus, the data logger on the embedded platform with 1 GB card and virtual machine platform data logger which uses the hard disk space is sufficient to log the transactions for many years.

Ninth, Offline analysis is one the major requirement of the SCADA data logger. Offline analysis means performing analysis on the acquired and stored data. This process involves the extraction of original data from the stored data using various mathematical, cryptography techniques etc. The offline analysis is important, because this analysis helps in the post forensic analysis of incidents and to find intrusion detections. For example, the attack documented in [5,17].In this case the offline analysis is not easy due to lack of data logging feature and investigation took years to find the reasons for the attacks. There are different ways the data can be accessed from the storage for offline analysis such as master terminal unit to poll the data logger and access the data to perform the offline analysis or directly access the data from the storage media (Flash Card and hard drives).

The SCADA data logger uses the following ways to access the stored data and perform the offline analysis. First the offline analysis on the embedded systems can be done in two methods: one method is to allow the control system master to poll the data logger and receive the information stored on the card. Retrieving stored transaction data is supported in two ways. First, the data logger supports using a dedicated serial port to all access to the stored transaction data. Second, the data logger supports retrieval of the transaction data over the same serial port being monitored. To support this retrieval process, the SCADA data logger is connected in the bump-in-wire configuration and assigned with a unique address. The address assigning for the data logger uses the unassigned MODBUS or DNP3 address. In both cases, the data logger monitors the transaction and checks the address. If a transactions is addressed to data logger then that transactions is not passed to the RTU/IED. Instead they are serviced by the data logger directly. At present, the SCADA data logger supports some of the queries from the master and retrieves data from the memory based on the query. Some of the queries supported by the SCADA data logger are *Retrieve_singlepacket, Retrieve_date and Retrieve_daterange etc. Retrive_singlepacket* is a query, which is used to retrieve a single packet based on the file name specified. Similarly, *Retrieve_date* and *Retrive_daterange* are set of queries which retrieve the data based on the date value specified. For example, if master address the SCADA data logger with Retrieve_date query, then the data logger which is continuously monitoring transactions and checking the address field does not pass the transactions to the slave. Based on the query, the data logger finds the transaction logged on the specified date in the query and returns the retrieved transactions back to master. Another method is removing the compact flash card directly from the FPGA and replacing it with a new card to log next transactions. The data on the card can be read using compact flash card readers using a personal computer and the offline analysis is performed on the retrieved data. As the data is in the encrypted form, it should be decrypted and verify the data using signing algorithm for data integrity. Second offline analysis on the virtual machine platform data logger, the data is logged to the internal hard drives on pc or external drives connected to the pc. In this case the offline analysis can be carried out by logging in to the PC as a user and can retrieve the required data for offline analysis. Thus, the SCADA data logger supports the offline analysis and helps in the post forensic analysis.

The last requirement of the SCADA data logger is to support the active analysis. At present the active analysis is not supported by the SCADA data logger, but this feature can be added to the data logger in future. To facilitate this feature in future, the data logger is connected in bump- in-wire configuration and is active with network. The data logger architecture contains a microprocessor which can be utilized to actively analyze capture transactions. Active analysis may be used to search transactions for intrusion signatures. This could be done by parsing transactions and comparing the transactions to signatures of transactions previously classify as invalid or illegal behavior. Active analysis may also be used to limit the number of packets which must be stored in the data logger's hard disk drive by classifying transactions which may be safely ignored and transactions which must be stored. Classifying transactions as ignorable would require methods similar to that used to classify transaction as illegal or invalid. Thus the SCADA data logger supports the Active analysis

Thus, all the requirements such data acquisition, log, store, offline analysis, harmless etc are satisfied by the SCADA data logger implemented on both the PC and embedded platforms.

4.5 SCADA Data Logger Architecture

The SCADA data logger is designed to work on virtual machine platform and embedded platform. It supports both MODBUS and DNP3 protocols. The data logger can be operated in two modes: Store-forward mode and fast forward-store mode. The implementation details and the architecture of SCADA data logger on both platforms: virtual machine platform and embedded platform is described in the following sections.

4.5.1 Embedded Platform

The architectural overview of the SCADA data logger on embedded platform is shown in the block diagram 4.4. The data logger is implemented using a Virtex 5 XUPV5-LX110T [32] field programmable gate array (FPGA) integrated circuit (IC) and 1 GB memory card. The FPGA is programmed with the Xilinx EDK [33] and C language is used for the development of software.

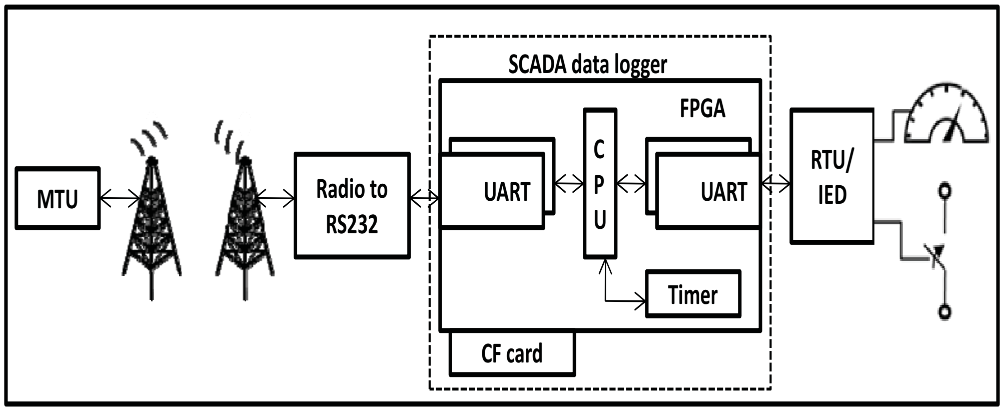


Figure 4.4 Embedded Platform SCADA Data logger

The FPGA contains processor (CPU), UARTs, Timer and Compact flash card (CF Card). Micro blaze is the soft core processor used as CPU in the implementation. It is a reduced instruction set computer (RISC) optimized for implementation on Xilinx FPGAs [34]. The micro blaze manages the process of logging captured transactions and can be used for the real time analysis and offline analysis of the logged transactions. The FPGA contains three serial UART circuits. Two of the serial UARTs` are used to monitor serial port traffic to and from the RTU/IED. The third serial UART is reserved to allow master nodes a direct connection to the data logger to retrieve stored transaction information. Timer on the FPGA is used to timestamp the network transactions when they are acquired from the UARTs. The timer on the FPGA always synchronizes with time clock on the master end. The transactions between the SCADA components are logged to 1 GB compact flash card available on the board. The transactions are stored to text files on the cards and a track of newly created files is maintained in a main log file. As future work, external memories like hard drives can be connected to the FPGA to increase the memory space for storage.

Figure 4.4 shows the SCADA data logger straddling a within a control system. The master node in Figure communicates with downstream nodes via radio. Use of radios to for point to point and point to multipoint communications is common in control systems since the radio save on wiring expense and allowed for communication over long distances. In control system shown the master node uses a RS-232 physical link to communicate with the RTU/IED. The radio connection carries the RS-232 traffic and delivers it to the RTU/IED. On the right hand side of the RTU/IED a sensor and a switch are shown. RTU commonly use integrated microcontrollers to monitor and control field devices such as sensors, switches, valves, and actuators.

The embedded platform version of the data logger is implemented using an FPGA is compact and portable. This version of the data logger can be deployed in remote sites easily, left untouched for years and it logs the network transactions efficiently.

4.5.2 Virtual Machine Platform

The virtual machine data logger is implemented as Linux process running in a virtual machine on a personal computer (PC) which hosts the HMI. This version of data logger uses the hard drive space on the host's PC hard disk to store the transactions. DNP3 and MODBUS protocols are supported by this data logger. The time stamping is based on the local system timing of the PC. The virtual machine data logger can be in upstream of the SCADA network to log the network transactions and provide the transactions related to response injection and denial of service attacks on MTU. The virtual machine platform data logger uses a single physical RS232 (UART) port to connect to the MTU and a virtual serial port to connect the HMI to the virtual machine running the data logger process. In this version, the CPU processes bytes received from the one UART and forwards to another UART. Generally, this way of implementation has higher processing speed and large memory to store the data. The architectural overview of the SCADA data logger on the virtual machine platform is shown in the figure 4.5.

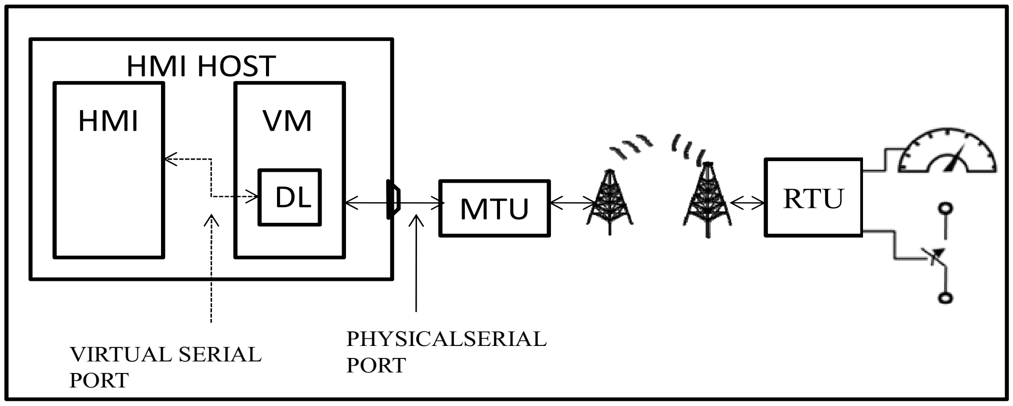


Figure 4.5 Virtual Machine Platform SCADA Data Logger

Figure 4.5 shows the data logger running as a Linux process on the HMI host PC. The physical port used to connect the data logger to the MTU and a virtual port to connect it to the HMI. The data logger operating on the HMI host PC logs the network transactions such as commands form the HMI to the and responses from the to the HMI. This version of data logger incurs less software implementation costs and the logged data is less prone to the attacks. The software implementation of this version data logger incurs less cost, because this version can be run as Linux process on the existing HMI host PC without any additional hardware As the HMI is protected safely with limited access to authorized persons, it is not easy for an attacker to access the logged data.

The virtual machine data logger can be placed in possible three locations in the upstream side to log the network transactions. First, the virtual machine data logger can be run as a Linux process on the PC hosting the HMI. This position of data logger can log network transactions related to commands from the HMI to the downstream elements and responses from the downstream elements to HMI. If an attacker gains access over the physical connections between the HMI and MTU and between the MTU and radio, he/she can still manipulate the responses from the downstream elements and commands from the HMI. Second, the virtual machine data logger version can be run as Linux process on another PC and can be placed in bump-in-wire configuration between the HMI host and MTU. In this case, still the data logger can log the all the network transactions efficiently. One possible chance for an attacker is to bypass the connection between the MTU and Radio and introduce any kind of attacks. Third, the virtual machine platform data logger can be run as Linux process on another PC and connected between the MTU and radio. This placement can also log the network transactions efficiently. The possible chances for an attacker are to gain access over the HMI PC or by pass the physical connection between the MTU and HMI host. Gaining access over the HMI, an attacker can issue the wrong commands to downstream elements and cause the malfunctioning of the control system. Hence, independent of the data logger placement in the SCADA network, the virtual machine platform SCADA data logger and embedded machine platform data logger logs the complete network transactions efficiently and logged data supports the post forensic analysis. Thus, the SCADA data logger supports two platforms: virtual machine and embedded platforms, supports the DNP3 and MODBUS Protocols.

4.6 SCADA Data Logger Operation Modes

The SCADA data logger can be operated in two modes: Store and Forward and Fast Forward and store. The flow charts of each mode are shown in the figure 4.6.

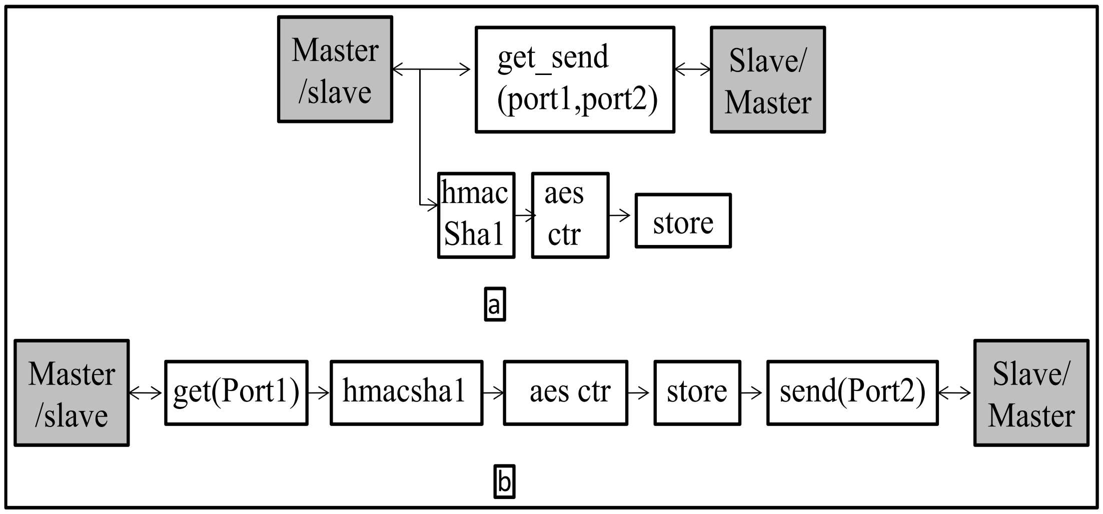


Figure 4.6 Flowcharts of Fast forward-store and store-forward modes

First the store-forward mode, in this mode, the Get function receives byte by byte from the port and produces a complete pdu (T_M).The transaction is time stamped while it is acquired from the UART. Once the pdu is received, it is concatenated with timestamp and nonce. Next the HMAC operation is performed on the concatenated value of the original transaction (PDU), timestamp and nonce and produces a tag value (MAC). Once the tag is obtained, it is combined with original transaction, timestamp and the result is encrypted with the Aes_Ctr. The output from the encryption function is stored on the compact flash card with the help of store function. After the process of storing the transaction to the card, the Send function sends the pdu through another port. The pdu retained in the data logger, while the operations of HMAC, Aes_Ctr and storage carried out. As a result, the latency increases and dependent on the length of message, time taken for cryptographic functions HMACSHA1 and AES_CTR and time taken to write the data to memory. This mode adds more latency to overall system and affects the normal function of the control system.

```
while (1){
        get(port 1)
        hmac_sha1
        aes_ctr
        store
        send(port 2)
}
```

Figure 4.7 Pseudo Code For Store and Forward Mode

Second the fast forward-store mode, in this mode, the Get_send function receives byte by byte from one port and simultaneously forwards the data to another port. The Get_send function is used to transmit and receive the data but also return the complete pdu after every transaction. Once the complete pdu is received, similar operations to store and forward mode such as concatenating the pdu with timestamp, nonce, performing the HMAC, Aes_Ctr, storing transactions to the compact flash is performed. The calculation of HMAC, Aes_Ctr and storing operations does not affect the operation of the control system and adds very less delay compared to store and forward mode.

```
while (1){
        get_send(port1,port2)
        hmac_sha1
        aes_ctr
        store
}
```

Figure 4.8 Pseudo Code For Fast forward-Store Mode

Thus the SCADA data logger can be operated in two modes. When compared in terms of latency, the fast forward and store is efficient method compared to store and forward mode. Because the fast forward mode adds less latency compared to store and forward mode and does not affect the normal function of the control systems.

4.7 Software Implementation of SCADA Data Logger

The SCADA data logger is implemented on virtual machine and embedded platforms, supports both SCADA communication protocols (MODBUS and DNP3). The software implementation of the SCADA data logger on virtual machine platform is implemented using a PC running Linux on Virtual machine environment, and the data logging process is implemented to run as a Linux process. The data logger in the virtual machine platform uses the internal hard disk connected to the PC to store the network transactions. Whereas the software implementation of the SCADA data logger on embedded platform is implemented using Embedded Development Kit (EDK) and Virtex- 5(XUPV5-LX110T) FPGA board. For details about the software development platform and syntaxes of the functions used in the implementation of the SCADA data logger on embedded platform refer to appendix B.

The software implementation of SCADA data logger on PC and embedded platform consists of two layers: application layer and hard ware abstraction layer. The Application layer is common for the both the platforms, whereas the hardware abstraction layer differs between the two platforms. The hard ware abstraction layer deals with the initialization of serial UARTS on the two platforms. On virtual machine platform, the UART is initialized using open ( ) function and runs at a baud rate of 9600.The functions read ( ) and write ( ) are used to receive and transmit data through the UARTs. On the embedded platform, the UART are initialized using the XUartNs550_Initialize, and operated at a baud rate of 9600. The XUartNs550_Recv and XUartNs550_Send are used to receive and transmit data through the UARTs. Table 4.2 lists various hardware abstraction layer and application layer functions used in the software development of the SCADA data logger.

The application layer consists of many functions to support both SCADA protocols (MODBUS and DNP3).The basic functions which are used in all other functions are Get_Byte and Send_Byte. The Get_byte uses read ( ) function on PC platform and XUartNs550_Recv on the embedded platform. This function receives data in order of byte size. The return value of Get_Byte is length of data receives. The other basic function is Send_Byte, which uses write ( ) function on PC platform and XUartNs550_Send on the embedded platform. This function is used to transmit the data through UART and the return is length of data transmitted.

Depending on the protocol supported by the data logger, various functions are defined in the application layer. First, the functions used by the data logger, when operating in the MODBUS ASCII MODE are Get_Modbus_Ascii, Send_Modbus_Ascii and Get_send_Modbus_Ascii. The Get_Modbus_Ascii function is used to receive the complete Modbus ASCII transactions from the network. This function calls the Get_byte function in a loop until the complete transactions is received. The return value of this function is MODBUS ASCII transaction.

Table 4.2 Various Data Logger Functions and Syntaxes

No	Functions	Syntax
1	XUartNs550_Initialize	`int XUartNs550_Initialize( XUartNs550* InstancePtr, u16 DeviceId)`
2	XUartNs550_Recv	`unsigned int XUartNs550_Recv ( XUartNs550 * InstancePtr, u8 * BufferPtr, unsigned int NumBytes )`
3	XUartNs550_Send	`unsigned int XUartNs550_Send ( XUartNs550 * InstancePtr, u8 * BufferPtr, unsigned int NumBytes )`
4	Get_byte	`int Get_Byte(int inport, char *byte)`
5	Send_Byte	`int Send_Byte(int outport, char byte)`
6	Get_Modbus_Ascii	`struct modbus_pdu *Get_Modbus_Ascii(int inport);`
7	Get_send_Modbus_Ascii	`Struct modbus_pdu * Get_Send_Modbus_Ascii(int inport, int outport)`
8	Send_Modbus_Ascii	`void Send_Modbus_Ascii(int outport, struct modbus_pdu *pdu);`
9	Get_Modbus_RTU	`struct modbus_pdu *Get_Modbus_RTU(int inport)`
10	Send_Modbus_RTU	`Void Send_Modbus_RTU(int outport, struct modbus_pdu *pdu)`
11	Sendlpdu	`void Sendlpdu(int outport, struct DNP3_lpdu *lpdu);`
12	Getlpdu	`int Getlpdu(int inport, struct DNP3_lpdu *lpdu);`

The Send_Modbus_Ascii is used to transmit the MODBUS ASCII transaction in to the network. This function calls the Send_Byte in a loop until the complete transaction is transmitted through UART and return value in NULL. The Get_Send_Modbus_Ascii function uses the both Get_Byte and Send_Byte functions to receive and transmit the transaction. Second, the data logger operating in MODBUS RTU mode uses the Get_Modbus_RTU and Send_Modbus_RTU. The Get_Modbus_RTU is used to receive data and the underlying function is Get_Byte and Send_Modbus_RTU uses Send_Byte for transmitting the data.

Similarly the DNP3 based data logger uses Sendlpdu and Getlpdu for transmitting and receiving operations. Thus, the software implementation of SCADA data logger on the virtual machine and embedded platform are explained.

4.8 Key Storage Mechanisms

The SCADA data logger uses cryptographic algorithms such AES with counter mode and HMAC SHA1.Both the algorithms involves secret keys, which should be protected safely. The design of SCADA data logger mainly aims at logging and uses the data for post forensic analysis. This work did not attempt to solve the key storage problem, but at present the keys are embedded in the software code running on the PC and embedded platforms. An attempt is made to find the various available methods for storing the keys safely and securely.

Storing keys is vital processes in any cryptographic applications. The safety and security of key storage have a major influence in the designing of cryptographic applications. Because various security services such as authentication, authorization, confidentiality and non-repudiation are dependent on cryptography and its applications. The SCADA data logger is implemented on virtual machine and embedded platforms. Several methods are available to protect the keys on both the platforms. When compared to embedded platform to the PC platform, the latter has higher processing power and mechanism to handle different processes at single time. In the case of PC based data logger, efficient key generation mechanisms can be used to generate the keys on the fly and can protect them safely in the secure databases. Due to limited resources on the embedded platform, the options are limited. Some of the key storage mechanisms for the embedded platforms are listed below. First method is to embed the long keys used for various cryptography techniques in the software code running on the FPGA [35].As the bit stream on the hardware is securely protected and prevents the stealing of keys by an attacker. Various bit stream protection schemes are available in market. For example, Xilinx based FPGA supports AES encryption of bit stream to accomplish security [36]. The key bits are stored in the nonvolatile memory and long life batteries are used to keep the memory active. The problem associated with this method is the secret key is fixed for long time.

Second method is to store keys of cryptography techniques directly in the nonvolatile memory available on the FPGA and supported with long life batteries [37]. The method is dependent on the life time of batteries used, failures of batteries etc. The keys can be accessed from the nonvolatile memory during run time. Third method is hardware implemented key storage, which allows the secure storage and high performance retrieval of the keys [38].In this method, a device is implemented on the top of a commercial off the shell (COTS) programmable hardware board (Celoxia RC1000) which is mounted on a Xilinx virtex E 2000 FPGA. This system contains two parts: controller and the store. The main function of the controller is to control the transfer operation of keys from board memory to FPGA memory. The board memory is used as buffer for holding the keys during the transferring. The store block is used for storing the keys safely and securely. This method helps to prevent the attacks at user level, root level and physical level. Thus any of above key storage mechanisms can be used for secure protection of the keys and can be integrated with the SCADA data logger more secure.

4.9 Summary

Thus this chapter includes the motivation for the design of the data logger, related works, explains the architecture of the SCADA data logger on the virtual machine and embedded platforms. The requirements of the data logger are listed, software implementation of the architecture on the embedded and virtual machine platforms, and operation modes of the SCADA data logger details and finally details about various available key storage mechanisms are explained in this chapter.

CHAPTER V

RESULTS AND ANALYSIS

5.1 Introduction

In this chapter the details about SCADA data logger validation and verification at MSU SCADA security Laboratory and in the Virtual machine desktop environment (Simulation environment) are explained. The latency measurements in different configurations and comparison studies between the various configurations, different modes of SCADA data logger and list of achieved SCADA data logger requirements are included in this chapter.

5.2 SCADA Data Logger Validation and Verification

The validation and verification of the SCADA data logger is done in several methods. The data logger supports two protocols MODBUS and DNP3 and works on both virtual machine and embedded platforms. The working of the data logger on both virtual machine platform and embedded platform is validated and verified using different setups.

First, the implementation and verification of SCADA data logger is done at MSU SCADA security Laboratory. The SCADA laboratory is equipped with two masters and five slave devices. The slave models that are configured in laboratory are a gas pipeline operation, an industrial blower (such as found in mining applications), a conveyor belt operation, a municipal water tower, and an oil/gas storage facility. This laboratory also contains two fully equipped HMI control stations currently running GE Proficy iFIX version

(4.5). The control stations receive both the wired and wireless signals. All stations in lab use non-traditional protocols and transmission medium (*e.g.*, 900 MHz radios and RS-232 links between master stations and slave PLC's). SCADAPacks hardware with an option for setting protocols as either MODBUS or DNP3 – two common control system protocols used in the US. For description about each component in the laboratory refer to Section 2.6.

Second verification and validation of the SCADA data logger is tested on VM ware desktop environment (Simulation Environment) where windows is the host operating system(Processor: AMD Turion @ 2 GHZ,3GB ram,250GB Hard disk space) and Linux runs as the guest operating system(Processor: AMD Turion @ 2 GHZ,3GB ram,8GB Hard disk space). On the guest operating system, both the master and slave terminals are initiated on the pc and working of SCADA data logger is validated. In this validation setup, the dummy packets resembling actual network traffic taken from MSU SCADA Security Laboratory are used to validate the working of the data logger. Thus the two testing environment for the validation and verification of the data logger are explained.

Table 5.1 shows the validation and verification of SCADA data logger using above mentioned environments. The SCADA data logger implemented on PC and embedded platforms supporting both MODBUS and DNP3 protocols is validated using the VMware Desktop environment. The MSU SCADA Security Laboratory is used to validate the SCADA data logger implemented on both virtual machine and embedded platform.

Table 5.1 Details About Validation of SCADA Data Logger on Different Environments

Validation Environment	Modbus				DNP3	
	Modbus ASCII		Modbus RTU			
	Embedded platform	PC platform	Embedded platform	PC platform	Embedded platform	PC platform
Simulation Environment	✓	✓	✓	✓	✓	✓
MSU SCADA security Laboratory	✓	✓	-	✓	-	-

Table 5.2 shows the validation and verification of the SCADA data logger in the MSU SCADA Security Laboratory. The virtual machine platform SCADA data logger operating in the two modes of the MODBUS i.e. MODBUS ASCII and MODBUS RTU works on all the 5 slaves and is able to log the transactions successfully.

Whereas the data logger implemented on the embedded platform supporting MODBUS ASCII is validated in the MSU SCADA Security Lab, but the implementation of RTU mode supporting embedded based platform data logger is not implemented and verified, due to limited resources such as implementation of timers and synchronizing them with each other etc.

Table 5.2 Details about Validation of SCADA data logger in MSU SCADA Security Laboratory

Control station	Modbus			
	Modbus ASCII		Modbus RTU	
	Embedded platform	PC platform	Embedded platform	PC platfor m
Gas pipeline	✓	✓	-	✓
Conveyor belt	✓	✓	-	✓
Industrial blower	✓	✓	-	✓
Municipal watertank	✓	✓	-	✓
Oil/gas storage	✓	✓	-	✓

Thus, the working of SCADA data logger based on MODBUS and DNP3 protocols are verified and validated using MSU SCADA Security Laboratory and VMware desktop Environment (Simulation Environment).

5.3 Testing Environment Configurations

The SCADA data logger is designed to log the communications between the master and slave. The data logger works on virtual machine and embedded platforms. It is compatible with the two SCADA protocols and MODBUS and DNP3.The data logger works in two modes: First one, Store and forward, in which the data logger receives the complete pdu from one end (Master) and performs the cryptographic operations on data, stores and then forwards it on another end (Slave). Second one fast forward and store, in this mode the pdu is not retained in data logger as long as the operations of cryptography and store process are performed. To study the impact of the SCADA data logger on the normal functioning of the control system, the latency of round trip communication is measured   with and without data logger in two modes.

The test environment used for the measuring the latency of round trip communication consists of the following equipment: Test PC running the VMware, where the host operating system is windows and Linux is the guest operating system, NULL modem cables for connecting the Laptop and the test module. The virtual machine platform SCADA logger and embedded based data logger is the test modules connected to the PC for measuring the latency in different configurations such as loop back, PC connected to embedded platform data logger ad PC connected to pc based data logger. The total environment communications   are operated at baud rate of 9600.

During testing three configurations are used in the measurement of the latency for the data logger. The three configurations used for the measurement of the roundtrip latency are Loopback configuration, PC to embedded platform data logger and PC to virtual machine platform data logger.

First, Loop back configuration is a basic configuration which resembles normal communication line between the master and slave in SCADA system. This configuration consists of test pc with VM ware and null modem. The SCADA data logger is not connected to test pc in this configuration. The configuration measurement provides a baseline for the comparison purposes. The loopback configuration is shown in the figure 5.1.

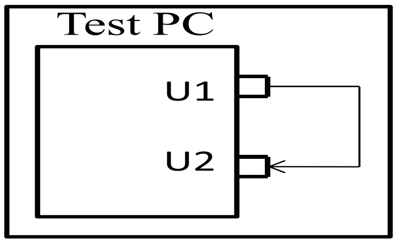


Figure 5.1 Loopback Configuration

Second configuration used is Test PC to embedded platform SCADA data logger. This configuration consists of PC running VM ware, null modem and the embedded platform data logger. The test PC and the data logger are connected with null modem cables. In Test PC to embedded platform SCADA data logger configuration, the data logger is operated in fast forward-store mode and store-forward mode for latency measurements. The PC to embedded platform SCADA data logger configuration is shown in the figure 5.2.

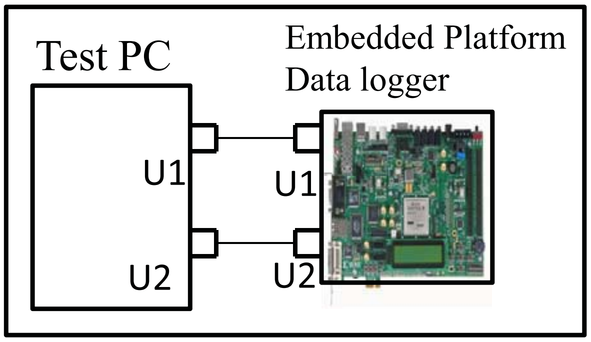


Figure 5.2 Test PC to Embedded Platform Data logger

The last configuration is Test PC to virtual machine platform data logger. This configuration consists of test PC running VM ware, null modem cables and virtual machine platform data logger. The test PC and the data logger are connected with null modem cables. In this configuration, the data logger is operated in fast forward-store mode and store-forward mode for latency measurements. The Test PC to virtual machine data logger configuration is shown in the figure 5.3

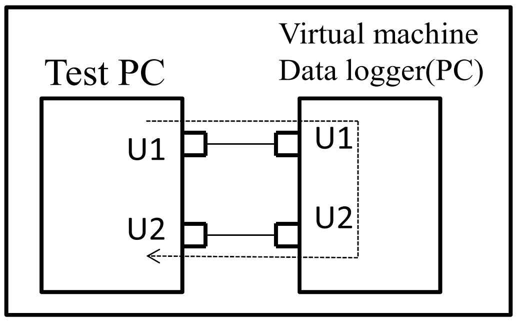


Figure 5.3 Test PC to Virtual Machine Platform Data Logger

In all the configurations, the PDUs are sent from one UART (U1) on the test PC and looped back to receive on the another UART (U2).Thus, the above configurations are used to measure the roundtrip latency and the obtained values is used to study the impact of the SCADA data logger on the control system functioning.

5.4 Measurement of Roundtrip latency

The latency is the time taken for a packet to reach destination from source and roundtrip latency includes both the time from source to destination and vice versa . The difference between those times gives the total round trip time. For example, let the t1 be the time at which a packet leaves the source and t2 be the time at which the packet is receives at source end from the destination. The difference between the t1 and t2 gives the total roundtrip latency i.e round trip latency = t2-t1.

The impact of the SCADA data logger on the control system is measured using the roundtrip time. To calculate the roundtrip latency, the local system time on the test PC with VM ware is used to time measurements. The time measurement starts when the first byte of the PDU was sent to the UART (U1) on the test PC and stopped when the first byte id received on the another UART(U2). For example, In the case of MODBUS ASCII, we have start identifier (colon) to indicate the start of the message and stop values

(CR, LF) to find the end of the message. Thus, we time measurement starts when the colon is sent to UART (U1) and the moment when colon is received on another UART (U2) the time measurement is stopped. The difference between start and stop values gives the roundtrip latency values. To analyze the impact of the SCADA data logger on the control system in terms of round trip latency, three test cases are used. The first two test cases use the VM ware Desktop environment (Simulation Environment) and last test case measurements are taken in the MSU SCADA security laboratory. The three test cases used and results obtained are explained in the following sections.

5.4.1 Test Case I

In the first test case, a set of 500 PDUs of different varying length is used. Each PDU is sent to UART (U1) on the test PC and time stamped when the first byte of the PDU is transmitted. When the first byte of the PDU is received back on another UART (U2), the data is also time stamped. The difference between the two timestamp values gives the roundtrip latency value. This method is used to make comparison between the different configurations such as loopback, Test PC to embedded platform data logger operating in fast forward –store and store-forward modes, Test PC to virtual machine platform data logger operating in fast forward–store and store-forward modes based on the roundtrip latency values. The results of the Test case 1 in different configurations are shown in the table 5.3

Table 5.3 shows the round trip latency values obtained for different configurations using the Test PC and following conclusions are made based on the data. First, the loop back configuration average 35ms and is the lowest of all the configurations, because in this configuration the data logger is not connected. So this configuration resembles the normal communication model between the MTU and with no delay. Thus, it is used as the baseline for the comparison study. Second, the comparison between the two platforms i.e. embedded platform data logger and virtual machine platform data logger operated in both the modes is studied.

The virtual machine platform data logger adds less delay compared to the embedded platform data loggers, because the virtual machine platform data loggers runs on a high processing HMI PC. The HMI PC has high processing power and resources compared to the embedded platform resources. Thus, the virtual machine platform data logger is efficient and adds less delay compared to the embedded version data logger.

The last comparison is made between the two modes of the SCADA data logger i.e. fast forward-store and store-forward mode on the virtual machine platform and embedded machine platform. The fast forward-store mode is faster compared to the store-forward mode and adds less delay to the control system. Because, in the store-forward mode, the PDU is retains in the data logger, until the operations of signing, encrypting and storing the data is done. Whereas, in the fast forward-store, the data logger processes the data bytes as received from one UART (U1) and forwards them to another UART (U2). When complete PDU is captured, the operation of signing, encrypting and storing the data is performed and these operations do not affect the normal functioning of the control system.

Table 5.3 Results of Test Case I on Three Configurations

Configuration (Mode)	Average Roundtrip Latency (ms)
Loop back	35.07
Test PC to embedded platform data logger (Fast Forward –store mode)	40.70
Test PC to embedded platform data logger (Store - forward Mode)	443.67
Test PC to virtual machine platform data logger (Fast Forward - store Mode)	39.86
Test PC to virtual machine platform data logger (Store - forward Mode)	420.89

So the data logger operating in the fast forward mode is faster than the store-forward mode. Thus, from the test case 1 makes a comparison study of roundtrip latency values between the different platforms and different modes of the data loggers. The Test case I results concludes that the fast forward-store mode faster than the store-forward mode and virtual machine platform data logger adds less delay compared to the embedded platform data logger.

5.4.2 Test Case II

The Second test case is used to study the effect of message length on the roundtrip latency in different configurations. For this test case, a set of 100 packets of varying lengths(10,30,50,100,150,200,230,250) are used. Table5.4 shows the roundtrip latency value in all the three configurations loopback, PC to embedded platform data logger and PC to PC based data logger in which the data logger can be operated in fast forward-store and store –forward mode.

Table 5.4 shows the roundtrip latency values on the different configuration with varying the length of the PDU. The length of PDU is varied from the 10 to 250. Based on the data obtained in the test case II, the following conclusions are made. First, the round trip latency values in the fast forward mode virtually remains constant and does not vary much with the increase in the length. Because in the fast forward-store mode, the data logger processes the bytes received from the UART (U1) and transmits to another UART (U2). As the time measurements in the test cases is taken when the first byte of the PDU is transmitted and received on the UARTs. So, the round trip latency time virtually remains constant.

Second, the round trip latency values in the store –forward mode is dependent on the length of the PDU and increases with increase in the length. In the store-forward mode, the PDU received from the UART (U1) by the data logger is not forwarded to another UART (U2) immediately. Once the complete PDU is captured, the operations such as time stamping, signing encrypting of data and storing are performed. After storing, the PDU is forwarded to another UART (U2).So as the length of the PDU increases, the time taken to do all the above operations also increases and results in increase in the roundtrip latency.

Table 5.4 Results Showing the Impact of PDU length on the Round trip Latency Time

Length	Configuration				
		Test PC to Embedded platform data logger		Test PC to virtual machine platform data logger	
	Loopback	Fast forward-store	Store-forward mode	Fast forward-store	Store-forward mode
10	30.85	42.63	358	33.67	325
30	30.08	40.13	408	35.98	389
50	33.37	44.55	645	36.09	590
100	33.11	40.13	980	35.77	914
150	31.81	48.06	1095	37.23	937
200	31.84	45.87	1136	35.23	968
230	32.49	44.23	1314	36.45	1197
250	30.06	43.56	1450	38.93	1366

Thus, the data logger operating in the store-forward mode is dependent on the length of the PDU and the roundtrip time increases with increase in the length.

5.4.3 Test Case III

The configuration setup used in the test case III is shown in the figure 5.4. The measurement results in this test case are taken in the MSU SCADA Security Laboratory to study the impact of the data logger on the model control system which resembles actual industrial setup. In this configuration, MTU, the test pc, data logger and slave RTU (gas pipeline control system and municipal water tower control system) is used to measure the roundtrip latency values. The dotted indicates the flow of the pdu sent from the MTU to test PC, test PC to RTU, response from RTU to test PC and from the test PC to MTU. First, the configuration is tested without any data logger to log the network transactions. This configuration includes the only MTU, Test PC and RTU. The test PC receives the commands from the MTU, timestamps when the first byte of the PDU is received and forwards the PDU to the RTU. Similarly the test PC receives the response from the RTU; timestamps when the first byte of the PDU is receives and forwards PDU to MTU. The difference between those two timestamps gives the roundtrip latency used as baseline for comparison studies.

Second, the data logger is included in the configuration and is connected between the test PC and RTU. The data logger is operated in the two modes of MODBUS i.e. MODBUS ASCII and MODBUS RTU.

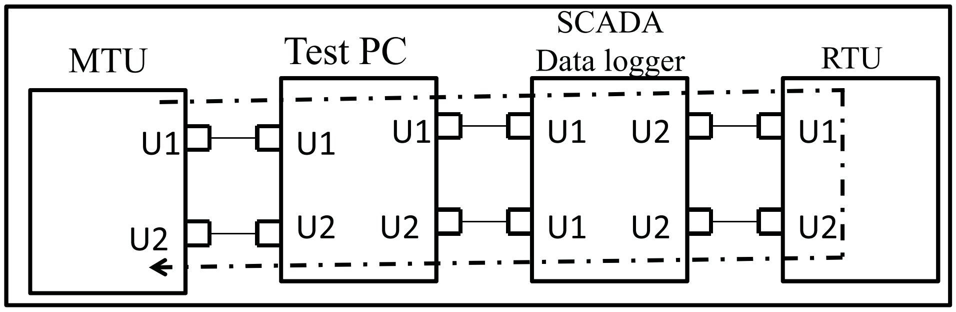


Figure 5.4 Test case Configuration for latency measurements in MSU SCADA Security Laboratory

At present, the DNP3 based data logger is not validated in the MSU SCADA Security Laboratory due to unavailability of DNP3 supporting master. When the SCADA data logger is working in MODBUS ASCII mode, the MTU and gas pipeline control system (RTU) are configured to MODBUS ASCII mode. Similar when the SCADA data logger is working in MODBUS ASCII Mode, the MTU and gas/oil storage control system (RTU) are configured in MODBUS RTU mode. In both modes, the MTU polls the slave connected to it for every 1 second.

Table 5.5 Results showing the impact of SCADA data logger tested in MSU SCADA Security Laboratory.

NO	Configuration	Roundtrip Latency
1	Basic configuration(No Data logger)	2.50 sec
2.	Configuration with Data logger operating in MODBUS ASCII Mode	2.80sec
3.	Configuration with Data logger operating in MODBUS RTU Mode	2.725sec

Table 5.5 shows the results taken in the MSU SCADA Security Laboratory by operating the data logger in both modes of the MODBUS i.e. MODBUS ASCII and MODBUS RTU modes. The table consists of average roundtrip latency values taken from the average of the 200 network transactions captured in the MSU SCADA Security Laboratory. The normal operation of the entire control system in the MSU SCADA Security Laboratory is not affected with the addition of the SCADA data logger, but it just adds minimal delay in the transmission of the transactions. Thus, the SCADA data logger does not affect the normal operation of the control system and satisfies its requirement being harmless.

5.5 Achieved SCADA Data logger Requirements

The SCADA data logger has different requirements which are introduced in chapter 1 and explained in detail in chapter 4.Table 5.6 lists the achieved SCADA data logger requirements. The SCADA data logger is designed to fit the legacy SCADA control system in bump-in-wire configuration and also supports the post forensic analysis. It is compatible to work on both platform and supports both SCADA protocols(MODBUS and DNP3).The latency values obtained in various configuration, validation of data logger in different environment(simulation and MSU SCADA Security

Laboratory) proves the data logger is harmless, but adds a minimal delay in the network. The SCADA data logger supports the offline analysis and lays foundation for active analysis in future. The time stamping of the network transactions helps in both the offline and active analysis. Finally, the SCADA data logger supports compact flash cards on embedded platform and hard drives in virtual machine platforms as storage devices. Thus the SCADA data logger meets all the requirements laid.

Table 5.6 Achieved SCADA Data Logger Requirements

SNO	Requirements	Achieved
1	Bump-in-wire retrofit	✓
2	Support Post forensic analysis	✓
3	Platform Compatible	✓
4	Protocol compatible	✓
5	Harmless	✓
6	Data Acquisition	✓
7	Log Data	✓
8	Store Data	✓
9	Offline analysis	✓
10	Support Active analysis	✓

5.6 Conclusions

Thus, this chapter includes various roundtrip latency measurements, comparison studies between different configurations such as loopback, PC- embedded platform data logger and PC to virtual machine data logger, lists the achieved requirements of the SCADA Data logger. Based on the results we conclude that fast forward-store mode adds less latency and have minimal impact on the normal function of the control system compared to the store-forward mode. Other conclusions are the virtual machine data logger is faster compared to embedded platform data logger, due to high processing of the virtual machine PC.

CHAPTER VI
CONCLUSIONS AND FUTURE WORKS

SCADA systems are used to monitor and control various critical processes in critical infrastructures such as water treatment plants, electricity generation, chemical processing etc. Failure of these critical infrastructure leads to financial loss and physical harm to the citizenry. Generally the SCADA network is poorly protected and makes system more vulnerable to various attacks such as response injection attack, command injection attack and denial of service attacks. The reasons various attack on the SCADA systems are unknown due to lack of data logging facility. Therefore, this thesis work aim at developing a retrofit SCADA data logger for the SCADA systems.

In this thesis, we have defined the need of data logging in the SCADA systems, implemented SCADA data logger architecture on virtual machine and embedded platform, listing the requirements a set of data logger requirements and explained how the requirements are achieved by the SCADA data logger.

The final conclusions of this thesis work are the SCADA data logger is implemented on the virtual machine platform and embedded machine platform. The data logger implementations are compatible to work with SCADA communications protocols such as MODBUS and DNP3.It is designed to operate in two modes: store-forward mode and fast forward–store mode. The cryptographic techniques such as HMAC SHA1 and AES operating in Counter mode (CTR) are used to provide the security to the logged data and the use the logged information in the post forensic analysis.

At present the SCADA data logger is limited to some basic functions such as logging, time stamping and protecting the data. Many features can be added to the data logger as future work to make it more efficient and compatible. At present, the SCADA data logger uses on RS-232 and RS-485 has physical layer. This can be extended to support Ethernet as future work.

The SCADA data logger developed is compatible to work on both the virtual machine and embedded platform. Due to limited resources and problems faced with the timers on the embedded platform, the SCADA data logger implementation on the embedded platform supporting MODBUS RTU is not successful. And validation of the DNP3 based data logger at MSU SCADA Security Laboratory is also not successful. So, this work can be extended to implement the MODBUS RTU supporting SCADA data logger and validation of DNP3 based data logger in laboratory.

In this thesis, the SCADA data logger does support the active analysis. To support active analysis in future, the SCADA data logger is connected in bump-in-wire configurations. But the active analysis is not implemented as a part of this work. So as a future work, an efficient intrusion detection system (IDS) can be integrated with the data logger to   capture packets and scan for the intruders at RTU (or MTU) online. This process helps in reducing the number of packets to be logged and helps in efficient utilization of the available memory. Thus active analysis feature can be extended to the SCADA Data logger.

The SCADA Data logger implemented in this thesis mainly aims at the capturing the network transaction in the SCADA network both in the upstream and downstream, protecting the logged information safely. To ensure safety of the logged transaction, various cryptographic techniques such as HMACSHA1 and AES-CTR are used and the keys used in these algorithms are embedded in the software code running on the data logger. As a future work, the efficient key storage mechanism, key generation methods can be investigated and integrate them with the SCADA data logger.

The storage mechanism implemented in this thesis is simple and logs one file for the one transaction, supports a few queries to retrieve the data back from the storage. As a feature work, the data base for logged transactions can be developed and add features such as querying the data base, backup the database on regular time basis etc. Efficient methods can be investigated.

Thus the conclusions and futures work of the SCADA Data logger implemented in this thesis.

REFERENCES

[1] MODBUS Protocol specifications. http://www.modbus.org/specs.php.Accessed June 01.2010 Accessed June 01,2010.

[2] R. Clarke. Practical modern SCADA protocols: DNP3, 60870.5and Related Systems. Newnes; 1st edition. September 2004.

[3] Electronics Industries Association, "EIA Standard RS-232-C Interface Between Data Terminal Equipment and Data Communication Equipment Employing Serial Data Interchange", August 1969, reprinted in Telebyte Technology Data Communication Library, Greenlawn NY, 1985, no ISBN

[4] Engineering Department, Electronic Industries Association, EIA Standard RS-485 Electrical Characteristics of Generators and Receivers for Use in Balanced Multipoint Systems, reprinted in Telebyte Technology "Data Communication Library" Greenlawn NY, 1985, no ISBN, no Library of Congress card number

[5] J. Slay and M. Miller, "Lessons Learned from the Maroochy Water Breach," IFIP Springer Boston, Vol. 253, pp73-82, 2007.

[6] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC:Keyed-Hashing for Message Authentication," RFC 2104,February 1997.

[7] NIST. Secure hash standard. Federal Information Processing Standard, FIPS-180-1, April 1995.

[8] J. Daemen and V. Rijmen. Answers to "New Observations on Rijndael". NIST AES website csrc.nist.gov/encryption/aes, August 2000.

[9] Lipmaa H., Rogaway P., Wagner D.: CTR-Mode Encryption, Public Workshop on Symmetric Key Block Cipher Modes of Operation. Oct. 2000, Baltimore, MD, http://csrc.nist.gov/encryption/modes/workshop1/

[10] American Gas Association, Cryptographic Protection of SCADA Communications;Part 1: Background, Policies and Test Plan, AGA Report No.12 (Part 1), Draft 5 (www.gtiservices.org /security/AGA12Draft5r3.pdf),April 14, 2005.

[11] Modbus Application Protocol Specification. http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf. Retrieved March 31, 2009

[12] MODBUS over Serial Line Specification and Implementation guide. http://www.modbus.org/docs/Modbus_over_serial_line_V1.pdf.Accessed June 01,2010.

[13] Curtis, K.A., “DNP3 Protocol Primer”, DNP Users Group, www.dnp.org/files/dnp3_primer.pdf), 2000, USA.

[14] R. Clarke. Practical modern SCADA protocols: DNP3, 60870.5and Related Systems. pages: Newnes; 1st edition.pg September 2004

[15] R. Clarke. Practical modern SCADA protocols: DNP3, 60870.5and Related Systems. pages: Newnes; 1st edition.pg September 2004

[16] T.Morris,K.Pavurapu,”A Retrofit Network Transaction Datalogger for SCADA control Systems”.IEEE SmartGridComm Track-Cyber/Physical Security and Privacy.-submitted.

[17] Abrams, M. and Weiss, 1. "Bellingham, Washington, Control System Cyber Security Case Study." National Institute of Standards. Information Technology Laboratory. August, 2007. Sampled on January 26, 2009. http://csrc.nist.gov/groups/SMA/fisma/ics/documents/Bellingham

[18] Freewave Radios. http://www.freewave.com/products/product-235.html Accessed June 01.2010

[19] Rockwell automation RSLOGIX 500 software. Available at http://www.ab.com/en/epub/catalogs/12762/2181376/2416247/1239758/2397811/ Accessed June 01, 2010.

[20] TELEPACE. Available at http://www.controlmicrosystems.com/media/page-bodyfiles/training/telepace/trainingmanuals/ Accessed June 01,2010.

[21] SCADAPACK PLCs and RTUs Available at http://www.controlmicrosystems.com/products/scadapack-plcs-rtus/

[22]IFIX PC monitoring software. Available at http://www.globalspec.com/FeaturedProducts/Detail/GEIntelligentPlatforms/Supervisory_Monitoring_and_Control_Software

[23] Das A.N., Popa D.O., Ballal P., Lewis F.L., “Data-logging and Supervisory Control in Wireless Sensor Networks”, in ACIS Int’l Journal of Wireless and Mobile Computing, 2007.

[24] B. Reaves, T. Morris. Discovery, Infiltration, and Denial of Service in a Process Control System Wireless Network. IEEE eCrime Researchers Summit. October 20-21, 2009. Tacoma, WA

[25] R. Chandia,J. Gonzalez,. T. Kilpatrick, M. Papa, S. Shenoi. Security Strategies for SCADA Networks. Pages117-131. Springer Boston. 2007.

[26] National Instruments. http://zone.ni.com/devzone/cda/tut/p/id/2946 accessed June 01,2010.

[27] B. Caswell, J. Bealeand, J. C Foster, and J. Faircloth, "Snort2.0 Intrusion Detection," Syngress, Feb. 2003.

[28] Control Microsystems, Inc.http://controlmicrosystems.com/, accessed June 01,2009

[29] J. Yang and F.-B. Sun. A comprehensive review of hard-disk drive reliability. In Proc. of the AnnualReliability and Maintainability Symposium, 1999.

[30] Kawaguchi, A., Nishioka, S., and Motoda, H. 1995. A flash-memory based file system. In Proceedings of the USENIX 1995 Technical Conference, New Orleans, Louisiana. 155–164.

[31] NERC CIP Standards. http://www.nerc.com/files/CIP-005-3.pdf. Accessed June 01,2010

[32] XUPV5-LX110 T Development System. http://www.xilinx.com/univ/xupv5-lx110t.htm. Acessed June 01.2010

[33] Xilinx Embedded Development Kit(EDK) http://www.xilinx.com/support/documentation/sw_manuals/edk10_ctt.pdf. Accessed June 01,2010.

[34] Xilinx MicroBlaze http://www.xilinx.com/tools/microblaze.htm. Accessed June 01,2010.

[35] R. Krueger. Using High Security Features in Virtex-II Series FPGAs.Xapp766 (v1.0),Xilinx, June 01, 2010. Available at http://www.xilinx.com/bvdocs/appnotes/xapp766.pdf.

[36] S. Drimer, "Authentication of FPGA bit streams: Why and how", In Proc. Of the International Workshop on Applied Reconfigurable Computing (ARC07), March 2007

[37] Xilinx, editor. Security Solutions Using Spartan-3 Generation FPGAs. Xilinx, San Jose, 2008.

[38] Alessandro Cilardo, Antonino Mazzeo, Luigi Romano, "An FPGA-based Key-Store for Improving the Dependability of Security Services," words, pp.389-396, 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems, 2005

APPENDIX A

CRYPTOGRAPHY BACKGROUND

Cryptography is the algorithmic method in which security goals are accomplished. The main goals of the cryptographic are privacy, integrity, authentication and non-repudiation. Many standards and algorithm are developed to achieve the above goals. First goal, privacy is the property of concealing the intent of the message. In particular, it means conceal the information from attackers while transmission of data takes places. Second goal, Integrity is the property of ensuring correctness in the absence of an actively participating adversary. In general, it means ensuring that a message is delivered safely from point A to Point B without any change in content. The property of the integrity can be obtained using one way hash functions. Third goal, authentication is process of validating the user or computer to ensure that it is what it claims. And the last goal of the cryptography is non–repudiation prevents either sender or receiver from denying a transmitted message.

The SCADA data logger implements the HMAC with SHA1 as underlying function and AES operating in counter mode (CTR) is used to secure the logged data. A brief description about each cryptographic functions used in the designing of the SCADA Data logger is explained.	The Advanced Encryption Standard is the NIST encryption standard The AES is more secure, reliable and efficient. It supports 128,196 and 256 key sizes. Generally AES can be operated in five modes: Electronic code book (ECB), Cipher-block chaining (CBC), Cipher feedback (CFB), Output feedback (OFB), Counter mode (CTR). First mode, electronic codebook is the simplest of all the encryption modes and suitable for short messages. Second mode, Cipher-Block Chaining (CBC) involves XOR operations of the cipher blocks with same single key. To produce the first block of cipher text, an initialization vector (IV) is XORed with the first block of the plain text. The initialization vector must be known to sender and receiver. Using this mode, the confidentiality and authentication property are achieved. Third mode, Cipher Feed Back Mode (CFB) is AES mode in which the input is considered as stream of bits. The major disadvantage of this mode is the error will propagates for the several blocks after the occurrence of error. Fourth mode, OFB operation is similar to CFB, except that the output of the encryption function is fed back to the shift register. In this mode, the error does not propagate in the transmission.

Last mode of the AES is the counter mode. In this mode, a counter equal to the plaintext block size is used. Typically, the counter is initialized to some value and incremented by 1 for each subsequent block. For encryption, the counter is encrypted and then XORed with the plaintext blocks to produce the cipher blocks. For decryption, the same sequence of counter is used, with each encrypted counter XORed with a cipher block to produce the original plain text. The major advantages of the counter mode are hardware efficiency, software efficiency; simplicity etc. The counter mode supports the property of parallelism which helps to increase the hard ware and software efficiency. The implementation of the AES_CTR is very simple, because it requires only implementation of the encryption algorithm and same algorithm can be used for the decryption also. Generally this mode is used in applications where high-speed network encryption is required.

HMAC is the cryptographic technique for calculating the message authentication code (MAC) involving a cryptographic hash function. Any iterative cryptographic hash function, such as MD5 or SHA-1 etc. may be used in the calculation of the HMAC. The strength of the HMAC depends upon the strength of the underlying cryptographic function, size of the key and size of the hash output. Then HMAC can be expressed as

$$\mathrm{HMAC}(K,m) = \mathrm{H}((K \oplus \mathrm{opad}) \parallel \mathrm{H}((K \oplus \mathrm{ipad}) \parallel \mathrm{m})). \qquad 2$$

In the above equation, H represents cryptographic function in which the underlying may be sha-1 or md5. K represents the secret key and b indicates the number of bits in a block. Opad and Ipad are the outer and inner padding used in the calculation of HMAC. The sha-1 function is the underlying function used in the calculation of HMAC for the logged data in the implementation of the data logger. The secure Hash Algorithm was developed by the National standards and Technology (NIST) and published as a federal information processing standard. The SHA1 algorithm takes input a message with a maximum length of less than 2^64 bits and produces output of 160-bit message digest. Big endian scheme is used in the SHA1 implementation. It is used in the wide applications such as, electronic funds transfer, software distribution, data storage. Thus, the SCADA data logger uses cryptographic algorithms such as HMACsha1 and AES-CTR to provide the data integrity and data confidentiality.

APPENDIX B

EMBEDDED PLATFORM FUNCTIONS

The Xilinx Embedded Development Kit (EDK) is a suite of tools and intellectual property (IP) that enables to design a complete embedded processor system for implementation in a Xilinx FPGA device. The EDK contains main tools such as Xilinx Platform Studio (XPS) and Software Development Kit (SDK). The XPS is the development environment for designing the hardware portion of embedded processors systems. The SDK is used for the designing of software application creation and verification. The Virtex-5 family of FPGAs offers a wide variety of the features such as high performance logic, serial connectivity, signal processing and embedded processing resources. The Xilinx XUPV5-LX110T is a versatile general purpose development board. This FPGA board contains board memory, connectivity interfaces like serial ports, Ethernet, 16x2 character LCD and 1 GB compact flash card.

The Software architecture of the SCADA Data logger consists of the following components: Micro blaze, Xps_Uart16550, Xps_timer, Xps_sysace and Xps_intc. The Microblaze is the software core processor and highly configurable. It is a Reduced Instruction Set Computer (RISC) and uses Harvard memory architecture. This processor runs at a frequency of 125Mhz.

The Xps_Uart166550 implements the hardware and software functionality of the national semiconductor 16550 UART. It also contains a 16bit programmable baud rate generator and independent 16 byte transmit and receive FIFOs. It can transmit and receive independently. Various UART functions such as XUartNs550_Initialize, XUartNs550_Recv, and XUartNs550_Send are used in the implementation of the UART in the SCADA data logger.XUartNs550_Intitialize function is used to initialize the uart component in the software implementation. *XUartNs550_Recv* and XUart550_send are the important functions plays vital role in receiving and transmitting the data through the uart.

Xps_timer/counter is the 32 bit timer module and have event generation, event capture capabilities. It can be operated in the count up and countdown mode. Xps_timer is used for the time stamping in the SCADA data logger. Various timer functions such as XTmrCtr_Initialize, XTmrCtr_Start, XtmrCtr_SetResetValue, and XTmrCtr_Stop, are in the SCADA data logger implementation. The XTmrCtr_Initialize function is used to initialize the timer component on the FPGA. XTmrCtr_Start and XTmrCtr_Stop are used to start and stop the timer and XtmrCtr_SetResetValue is used to set the reset for the timer.

Xps_sysace is also known as XPS system ACE interface Controller. It uses multiple interfaces such as Compact Flash, MPU and JTAG. In SCADA data logger compact flash card is used to log the transactions. To access the compact card for reading and writing purposes, XilFats Fat file System access library is used [53]. This library works for Fat 12, Fat 16 or Fat 32 files. Sysace_fopen, sysace_fread, sysace_fwrite and sysace_fclose are various fat file functions used in the implementation of SCADA data logger. First function, Sysace_fopen is file operation used to open the file on the flash device. It operates in two modes: write and read mode. In read mode, the function opens a named file stored on the flash drive. If named file does not exist on the device, zero (0) is returned. In write mode, It creates a new file on the device with given file name. Second function, Sysace_fwrite function is used to write the data from a pre allocated buffer to a specified file on the compact flash and it returns the number of characters written to file. Third function, Sysace_fread function is defined to read the data from the compact flash card to pre –allocated buffer and it returns the number of characters read from the file. Last function, sysace_fclose is used to close the opened file. Generally, the sysace_fwrite and sysace_read functions are followed by this function to prevent the corruption of the files on the device.

Xps_intcis the Interrupt controller used to control the interrupts from the various peripheral devices. The major function of the interrupt controller is to capture the interrupts from various peripheral devices connected to processor, acknowledge the devices and enable the interrupt conditions In SCADA data logger, only timer (XTmrCtr) is connected to interrupt controller and operated in the interrupt mode. XIntc_Initialize, XIntc_Connect, XIntc_Start, XIntc _Stop are the different interrupt function used in the data logger implementation. First function, XIntc_Initialize is function to initialize the interrupt controller on the embedded system. Second function, XIntc_Connect is used to establish the connection between the interrupt source and the associated handler dependent on the interrupt. Third function, XIntc_Start is used to start the interrupt controller by enabling the controller output to the processor. It is operated in the two modes: simulation and real mode. In the simulation mode, only simulation of the interrupts is enabled whereas in the real mode the actual hardware interrupt are also enabled. Last function, XIntc _Stop is used to stop the interrupt controller by disabling the controller output to the processor. After using this function is called, the interrupt controller does not handle any more interrupt from the peripheral devices

The various UART used in the implementations of the SCADA Data Logger are the following: XUartNs550_Initialize,XUartNs550_SetBaud,XUartNs550_msetLineControlReg, XUartNs550_Recv, and XUartNs550_Send.The syntax of each function is given below:

XUartNs550_Initialize:

Syntax:

int XUartNs550_Initialize (XUartNs550 * *InstancePtr*, u16 *DeviceId*)

InstancePtr: Pointer to the XuartNs550 instance,

DeviceId: The unique id of the device controlled by the XUartNs550 instance

XUartNs550_SetBaud:

Syntax:

void XUartNs550_SetBaud (u32 *BaseAddress*, u32 *InputClockHz*, u32 *BaudRate*)

Base Address: Contains the base address of the Device

Input Clock: The frequency of the input clock.

Baud Rate: The baud rate to be set

XUartNs550_Recv:

unsigned int XUartNs550_Recv (XUartNs550 * *InstancePtr*, u8 * *BufferPtr*, unsigned int *NumBytes*)

Instance Ptr: The pointer to the UART instance

BufferPtr: Pointer to buffer for data to be received into

Numbytes: The number of bytes to be received.

XUart550_Send:

Syntax:

unsigned int XUartNs550_Send (XUartNs550 * *InstancePtr*, u8 * *BufferPtr*, unsigned int *NumBytes*)

Instance Ptr: The pointer to the UART instance

BufferPtr: Pointer to buffer for data to be sent

Numbytes: The number of bytes to be received.

The various timer functions are used in the implementation of the SCADA Data logger. The timer is used to timestamp the data that is from the UARTs. The functions used for implementation of the timer are the following: XTmrCtr_Initialize, XTmrCtr_Start: XtmrCtr_Stop, XTmrCtr_SetOptions, and XtmrCtr_SetResetValue.

XTmrCtr_Initialize:

Syntax:

int XTmrCtr_Initialize (XTmrCtr * *InstancePtr*, u16 *DeviceId*)

Instance ptr : Pointe to the XTmrCtr instance

DeviceId : The unique id of the device controlled by XTmrCtr instance. The value can be obtained from xparameters file.

XTmrCtr_Start:

Syntax:

void XTmrCtr_Start (XTmrCtr * *InstancePtr*, u8 *TmrCtrNumber*)

where:

Instance Ptr : Pointer to the XTmrXtr instance

TmrCtrNumber: The number of the timer counter on which the device have to operate. Generally each device may have multiple timer counters.

XtmrCtr_Stop

Syntax:

Void XTmrCtr_Stop (XTmrCtr* *InstancePtr*, u8 *TmrCtrNumber*)

Where:

InstancePtr: The pointer to the XTmrCtr Instance.

TmrCtrNumber: The number of the timer counter on which the device have to operate. Generally each device may have multiple timer counters.

XTmrCtr_SetOptions

Syntax:

u32 XTmrCtr_SetOptions (XTmrCtr * *InstancePtr*, u8 *TmrCtrNumber*)

Where:

InstancePtr: The pointer to the XTmrCtr Instance.

TmrCtrNumber: The number of the timer counter on which the device have to operate. Generally each device may have multiple timer counters.

XtmrCtr_SetResetValue

Syntax:

Void XtmrCtr_SetResetValue (XTmrCtr * *InstancePtr*, u8 *TmrCtrNumber*, u32 *ResetValue*)

Where:

InstancePtr: The pointer to the XTmrCtr Instance.

TmrCtrNumber: The number of the timer counter on which the device have to operate. Generally each device may have multiple timer counters.

Reset Value: The value to reset the timer counter.

The XilFats file system library provides the read/write access to files stored on the Xilinx compact flash drive. This library works with Fata16 on Xilinx FPGA. The following are the functions used to perform the various operations: *Sysace _fopen,* Sysace_fread, Sysace_fwrite and Sysace_close.

Sysace _fopen

This is sysace file operation to open the file stored on the flash device. It operates in two modes: read mode and write mode. In read mode, the function opens a named file stored on the flash drive. If named file does not exist on the device, zero (0) is returned. In write mode, It creates a new file on the device with given name.

Syntax:

void *sysace_fopen (const char *file, const char *mode)

Sysace_fread:

This procedure is used to read data from file opened by sysace_fopen function in to pre allocated buffer. The function returns the number of characters read from file. The value of the size is one(1).The size of the buffer must be at least equal to the value of count.

Syntax:

int sysace_fread (void *buffer, int size, int count, void *file)

Sysace_fwrite

This function is used to write the data from pre- allocated buffer to compact flash drive with specified file name. The value of the size is one (1).The size of the buffer must be at least equal to the value of count.

Syntax:

int sysace_fwrite (void *buffer, int size, int count, void *file)

Sysace_close

This function is used to close the open file. It also synchronizes the buffer memory to cache memory. Generally, the sysace_fwrite and sysace_read functions are followed by this function to prevent the corruption of the files on the device. It returns an integer value of 0 on success or -1 on failure of function.

Syntax:

int sysace_fclose (void *file)

The XIntc is used to control the interrupt from the various peripherals and process them. The main functions of the interrupt controller is to capture, acknowledge and enable the interrupts. The various Interrupt function used in the implementation of the SCADA data logger are the following: XIntc_Initialize, XIntc_Connect, *XIntc*_Start XIntc _Stop.The syntax of each function is given below.

XIntc_Initialize

This interrupt function is used to initialize a specific interrupt controller instance.

Syntax:

int XIntc_Initialize (XIntc * *InstancePtr*, u16 *DeviceId*)

InstancePtr: Pointer to the XIntc instance

DeviceId: The Unique id of the device controlled by XIntc instance.

XIntc_Connect

Syntax:

int XIntc_Connect (XIntc * *InstancePtr*, u8 *Id*, XInterruptHandler *Handler*, void * *CallBackRef*)

InstancePtr: The pointe to the XIntc instance.

Id: This contains the ID of the interrupt source.

Handler: Interrupt Handler

CallBackRef: The instance pointer of the connecting driver.

XIntc_Start

This function starts the interrupt controller by enabling the output from the controller to the processor. This function operated in two modes: simulation mode, in which only simulation of the interrupts is enabled and in the Real Mode the hardware interrupts are also enabled. The return value for the function is either XST_SUCCESS or XST failure. Generally this function is called only after the initialization of the Interrupt controller.

Syntax:

int XIntc_Start(XIntc * InstancePtr,u8 Mode)

InstancePtr: pointer to the XIntc instance.

Mode: Either real or simulation.

XIntc _Stop:

This function is used to stop the interrupt controller by disabling the output. After this function is called, the interrupt controller does not initialize any more interrupts. The return value of the function is Null.

Syntax:

Void XIntc_Stop(XIntc * InstancePtr)

Where InstancePtr : pointer to the XIntc controller.