BLOOM FILTER BASED INTRUSION DETECTION FOR SMART GRID

A Thesis

by

SARANYA PARTHASARATHY

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2012

Major Subject: Electrical Engineering

BLOOM FILTER BASED INTRUSION DETECTION FOR SMART GRID

A Thesis

by

SARANYA PARTHASARATHY

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Chair of Committee,	Deepa Kundur
Committee Members,	Raffaella Righetti
	Erchin Serpedin
	Tiffani L. Williams
Head of Department,	Costas N. Georghiades

May 2012

Major Subject: Electrical Engineering

ABSTRACT

Bloom Filter Based Intrusion Detection for Smart Grid.

(May 2012)

Saranya Parthasarathy, B.Tech., National Institute of Technology, India

Chair of Advisory Committee: Dr. Deepa Kundur

This thesis addresses the problem of local intrusion detection for SCADA (Supervisory Control and Data Acquisition) field devices in the smart grid. A methodology is proposed to detect anomalies in the communication patterns using a combination of n-gram analysis and Bloom Filter. The predictable and regular nature of the SCADA communication patterns is exploited to train the intrusion detection system. The protocol considered to test the proposed approach is MODBUS which is used for communication between a SCADA server and field devices in power system. The approach is tested for attacks like HMI compromise and Man-in-the-Middle.

Bloom Filter is chosen because of its strong space advantage over other data structures like hash tables, linked lists etc. for representing sets. The advantage comes from its probabilistic nature and compact array structure. The false positive rates are found to be minimal with careful choice of parameters for Bloom Filter design. Also the memory-efficient property of Bloom Filter makes it suitable for implementation in resource constrained SCADA components. It is also established that the knowledge of physical state of the power system i.e., normal, emergency or restorative state can help in improving the accuracy of the proposed approach.

DEDICATION

To My Family

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Deepa Kundur, for providing me an opportunity to work under her and also for her excellent guidance and support throughout the course of this research.

I would like to thank my committee members, Dr. Raffaella Righetti, Dr. Erchin Serpedin, Dr. Tiffani L.Williams, for agreeing to be on my thesis committee and providing me valuable feedback.

I would also like to thank the department faculty and staff for making my time at Texas A&M University a great experience. I would like to thank my parents and family for providing me with the best of opportunities and having faith in my abilities. I wish to thank all my friends who made my stay away from home a memorable experience.

TABLE OF CONTENTS

Page

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

1. INTRODUCTION

A smart grid is defined as “the integration of real-time monitoring, advanced sensing, and communications, enabling the dynamic flow of both energy and information to accommodate various forms of supply, delivery, and use in a secure and reliable electric power system, from generation source to end-user” [1] (according to the definition by North American Electric Reliability Corporation). The main motivations behind smart grid are to achieve better control over the delivery of electricity to individual consumers thereby optimizing energy usage, enabling dynamic pricing of electricity and promoting renewable energy sources. To enable better control, information flow is two-way which will enable an electric utility to monitor the energy usage inside customer’s premises. Smart meters are being newly installed at households for this purpose. Another major change for the purpose of smart grid is the integration of Information Technology (IT) systems with the existing power system components for better monitoring and control.

For example, SCADA (Supervisory Control and Data Acquisition Systems) which are one of the major components of the power system are becoming information technology(IT) enabled with components like sensors, IEDs (Intelligent Electronic Devices) etc. to provide real-time measurements to the control center. These measurements are used for major functions like contingency analysis, corrections for

This thesis follows the style of *IEEE Transactions on Automatic Control.*

real and reactive power dispatch etc. It is common to have corporate IT networks connected to the SCADA networks for the collection of production data as shown in Figure 1.

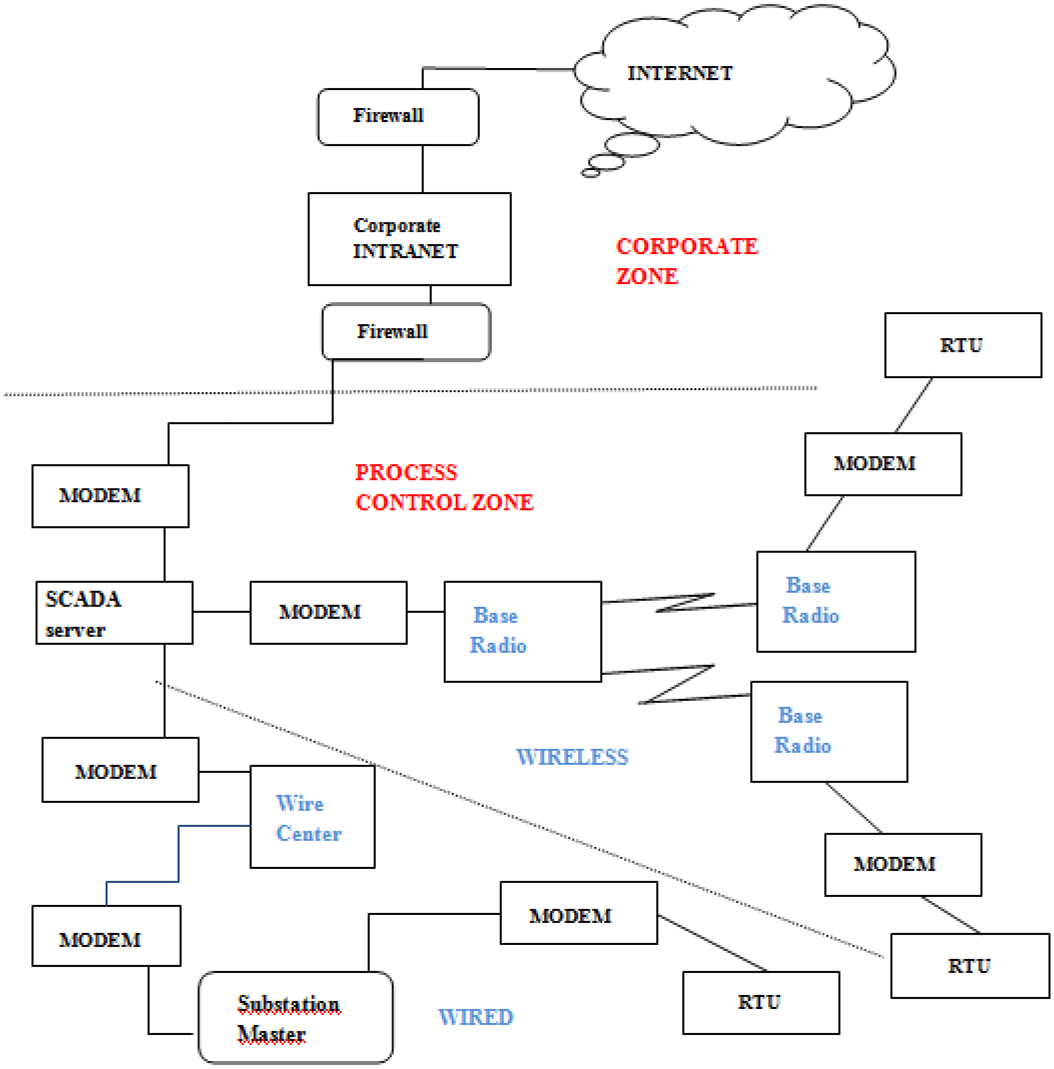


Figure 1 SCADA System Interconnection with Corporate Networks

Due to the increase in the IT components in the grid, the number of potential electronic access points increases which can be targeted by the attackers. If any vulnerability in newly installed smart meters, communication networks, or SCADA systems can be exploited by the attackers, the entire grid can be compromised. For example, the consequences of affecting a major component in a power system by gaining access to a single component were demonstrated by Idaho National Labs in 2007. This demonstration showed the capability of an attack targeting a generator by gaining access to an associated control system. This eventually led to destabilization of a generator bringing it to a halt. Another recent example of how a failure in any part of the grid can have cascading effects is the 2003 Northeast power blackout which affected nearly 50 million people [2]. Power was restored in most affected areas within 48 hours, but it was not restored in few places for up to a week. The propagation of the failure was very fast as it took only seven minutes for it to sweep across the region. Though this is not a cyber attack, it shows the impact of cascading failures on the grid. Cyber attacks can cause similar cascading failures if they target certain key elements of the power system.

In response to the increasing threats and vulnerabilities in the power grid, various security measures have been proposed. These include

- Encryption of network traffic.
- Better authentication mechanisms like using tokens and biometric measures.
- Not using default passwords in field devices like RTUs (Remote Terminal Units), PLCs (Programmable Logic Controllers) etc.
- Increased monitoring of network activities for detection of attacks, access logging.

- Vulnerability assessment, penetration testing of power system components.

Despite all these measures, there are still a lot of vulnerabilities which can be exploited by an attacker. One main reason is that power systems, like most other critical infrastructures were not initially designed with cyber security as an important criterion in mind. Other reasons include the complex architecture of the grid and lack of security culture among the grid operators.

1.1 Cyber Security in SCADA Systems

SCADA systems are one of the major components in the power systems. These are responsible for distributed monitoring, control, collection and analysis of real-time data over a wide area. The main components of a SCADA system are SCADA server and field devices like PLCs, RTUs and IEDs as shown in Figure 1. The interconnection of SCADA network with the enterprise or corporate networks is the weakest link in SCADA system security. Compromised SCADA systems will have a direct impact on the underlying physical system, i.e., the power system. For instance, even changing the status of a single circuit breaker from ‘open’ to ‘close’ can cause considerable damage. This can range from equipment damage to impacting personnel safety.

The SCADA components have certain vulnerabilities which can be exploited to compromise them. They can be mainly classified into two kinds as follows:

System-level: As SCADA systems are connected to corporate networks, compromising the corporate networks and the associated firewalls is a means of gaining access to SCADA components. Even without compromising them, an attacker can gain access to certain SCADA components like field devices which are equipped with Blue tooth and

Wi-Fi technologies. These are examples of vulnerabilities which arise by design. One solution to handle them is to disallow remote access to field devices. But it is not a viable option as this will affect the desired functionality. Suppose individual field devices are made secure and robust, interconnection of individually secure devices or systems can still give rise to certain vulnerabilities. For example, a nuclear power plant was automatically shut down due to the rebooting of a control system workstation though the workstation and control system were not on the same network [3]. The individual system-level and interconnection vulnerabilities can be addressed by some existing off-the-shelf commercial security solutions. But such solutions may not be suitable because of the following reasons:

- There are some SCADA specific operating systems and applications which may not be interoperable with existing COTS (commercial off-the-shelf) cyber security solutions.
- Some of the SCADA field devices do not have the sufficient computing resources for enabling security.

Protocol-level: Protocols used for communication in SCADA systems include DNP3, MODBUS, etc. These do not have even primitive authentication, integrity checks or encryption options. This is because they were initially designed only for serial communication but later modified and enhanced to run over TCP/IP, Ethernet, etc. Hence there are vulnerabilities from two sources, generic vulnerabilities from TCP/IP and the vulnerabilities specific to the protocol used (MODBUS, DNP3, etc).

These vulnerabilities can be exploited to launch various kinds of attacks as discussed below.

Flooding/Denial of Service: This attack can be launched in two ways, blocking data flow through SCADA communication networks or by affecting the timeliness by delaying the data. This can be achieved by flooding a network with unwanted packets. If the SCADA communication channel is flooded with unwanted packets, the field devices will try to execute the commands received from all the packets which may result in the exhaustion of their resources. Hence the devices will stop responding to the legitimate packets in course of time and data flow gets affected.

Man-in-the-middle attack: Due to lack of integrity checks in SCADA protocols, attackers can intercept the communication between field devices and SCADA server and modify the MODBUS request and response messages according to their will. These forged messages can be used to execute commands in a PLC and cause any damage to the underlying physical system. Examples include unwarranted opening or closing of circuit breakers, switches etc.

Replay attacks: Due to the lack of time-stamps in SCADA protocols, attackers can modify packets and transmit them later for their malicious acts. For example, one type of attack is to capture packets during normal operation at the control center and play them back to the operator indicating normal behavior while the attacker continues to send malicious commands to the field devices.

Vulnerabilities in SCADA systems have been exploited in the past and attacks have been launched against critical infrastructure as discussed in the examples below:

Stuxnet: Stuxnet worm was discovered in the summer of 2010. It started by exploiting multiple zero-day vulnerabilities on PCs running Microsoft Windows. Once it enters a computer system through shared network files or a USB, it looks for a specific Siemens SCADA program. If this program is running in the system, it searches for a particular industrial equipment configuration. If this configuration is found, then read/write requests to PLCs are intercepted and manipulated while reporting normal functioning to the operator. The modifications to the PLC code are hidden using root kits. As these PLCs are responsible for controlling the spinning speeds of centrifuges, this attack caused huge damage to the rotating centrifuges [4]. It has been widely believed that Stuxnet is aimed at causing heavy damage to Iran's uranium enriching facility at Natanz.

Aurora: The Aurora demonstration performed by the Idaho National Laboratory (INL) and shown on CNN used a cyber-attack to destroy a diesel generator. They used dial-up modems to attack diesel generator [5].

Maroochy WasteWater Wireless SCADA attack: This is the case of a disgruntled ex-employee using wireless equipment to access the SCADA system, by issuing radio commands to sewage equipment which caused large quantities of raw sewage to flow into local parks, rivers etc [5]. The sewage system experienced multiple serial faults during this incident like pumps not running when they should have been, alarms not being properly reported to the central computer, problems in communication between the central computer and the field devices (i.e.) pumping stations. This is an example of an insider attack which included stealing equipment, issuing unwarranted radio commands, using false network address, sending false instructions and disabling alarms.

Such attacks on SCADA system components have considerable impact on critical infrastructures. Hence security is an important concern for SCADA systems or industrial control systems associated with critical infrastructures.

1.2 Intrusion Detection

Cyber intrusion is a means of compromising a system by performing unauthorized activities. To reduce such intrusions on SCADA systems, certain measures have been suggested like isolating SCADA from corporate networks, encrypting SCADA data and enforcing validated passwords, introducing redundancies for critical nodes etc. Despite these measures, there is still a need to have an intrusion detection and recovery mechanism which is evident from the recent attacks on industrial control systems discussed in the Section 1.1. There are some challenges in developing these SCADA specific intrusion detection systems as they different from the conventional IT system in the following ways:

- Timeliness and availability at all times is critical, hence the CIA (Confidentiality-Integrity-Availability) in IT system becomes AIC (Availability-Integrity-Confidentiality) in SCADA systems.
- Availability of limited memory and computing resources in field devices like PLCs, RTUs
- Logic inside SCADA has a direct impact on the physical system it controls.

- Attack on a SCADA system can have cascading effects on the underlying physical system. There should be protection measures to prevent such cascading failures and aid in graceful degradation of the physical system.

The design of an intrusion detection system for SCADA should take into consideration the above differences.

1.3 Contributions of This Thesis

The main contributions of this thesis are as follows:

1. A Bloom Filter based intrusion detection approach for resource constrained SCADA components in the smart grid is proposed. Bloom Filter is a data structure which represents a set of data in a concise form and hence suitable for representing the set of normal traffic patterns in power system SCADA communication concisely. Also as it has low memory requirements, it is suitable for localized intrusion detection in SCADA components like RTUs, PLCs which have limited computational resources.
2. The proposed approach has been tested for SCADA communication traces using MODBUS protocol which is used for communication between a SCADA server and devices like RTUs and PLCs.
3. Bloom Filter is compared against another data structure, the standard hash table. It is found that look ups are of order O (1) in Bloom Filter where as it is not the case with standard hash table implementation. In hash tables, there is an extra overhead for searching through a linked list for look ups. Hence, in comparison

with hash tables, Bloom Filter is found to be efficient in performing look ups for identifying normal behavior during intrusion detection.

4. The intrusion detection approach is tested with and without the knowledge of the current state of the underlying physical system. It is found that consideration of the current state of the physical system, (i.e. power system) along with the information from cyber components, (i.e., parameters extracted from SCADA network traffic) will be helpful in improving the quality of the proposed intrusion detection system.

1.4 Organization of This Thesis

In Section 2, we provide an overview of the components involved in power system SCADA communication and the MODBUS protocol used in SCADA communication. Also we explain the intrusion detection techniques in general and discuss the existing SCADA specific intrusion detection approaches. In Section 3, we discuss the SCADA architecture, threat model and attack scenarios considered for our proposed intrusion detection framework. We compare the two approaches, hash-table and Bloom Filter for SCADA intrusion detection and explain the advantages of using Bloom Filter approach. We also discuss how the various design parameters of Bloom Filter are chosen. Then we explain how the correlation between the states of the power system with the SCADA network traffic can be leveraged to improve the intrusion detection mechanism. In Section 4, we present the results obtained for Bloom Filter based approach by using training data for both normal and emergency states of the power system. In Section 5, we provide an assessment of the results and present directions for further improvements and future work.

2. BACKGROUND

This section provides an overview of the components involved in the power system SCADA communication in Section 2.1. This communication involves SCADA protocols and one of these protocols named MODBUS is explained in detail in Section 2.2. Then the general intrusion detection techniques and a literature review of the existing SCADA specific intrusion detection approaches are discussed in Section 2.3.

2.1 Communications in SCADA for Power Systems

The major components in Power System SCADA are as follows: a central computer server called as the SCADA server or SCADA master, field data interface devices like Programmable Logic Controllers (PLCs), Remote Terminal Units (RTUs) and Intelligent Electronic Devices (IEDs), a communication system to transfer data between the field devices and SCADA server and a user interface called Human Machine Interface (HMI). RTUs convert data from devices like sensors, switches, transmitters into a form which can be transmitted over the communication channel. PLCs store instructions for the field devices in the form of procedures. They act on the inputs from RTUs and perform actions based on time, events or a particular sequence. For example, they are responsible for opening or closing a sequence of circuit breakers under emergency conditions. The protocols used in the communication system include MODBUS, DNP3 which follow a master-slave technique for communication. The SCADA servers process the information received from PLCs, RTUs and present them in a form which can be interpreted by the human operator. This processed information is displayed in a User Interface to the operator. These are shown in Figure 2.

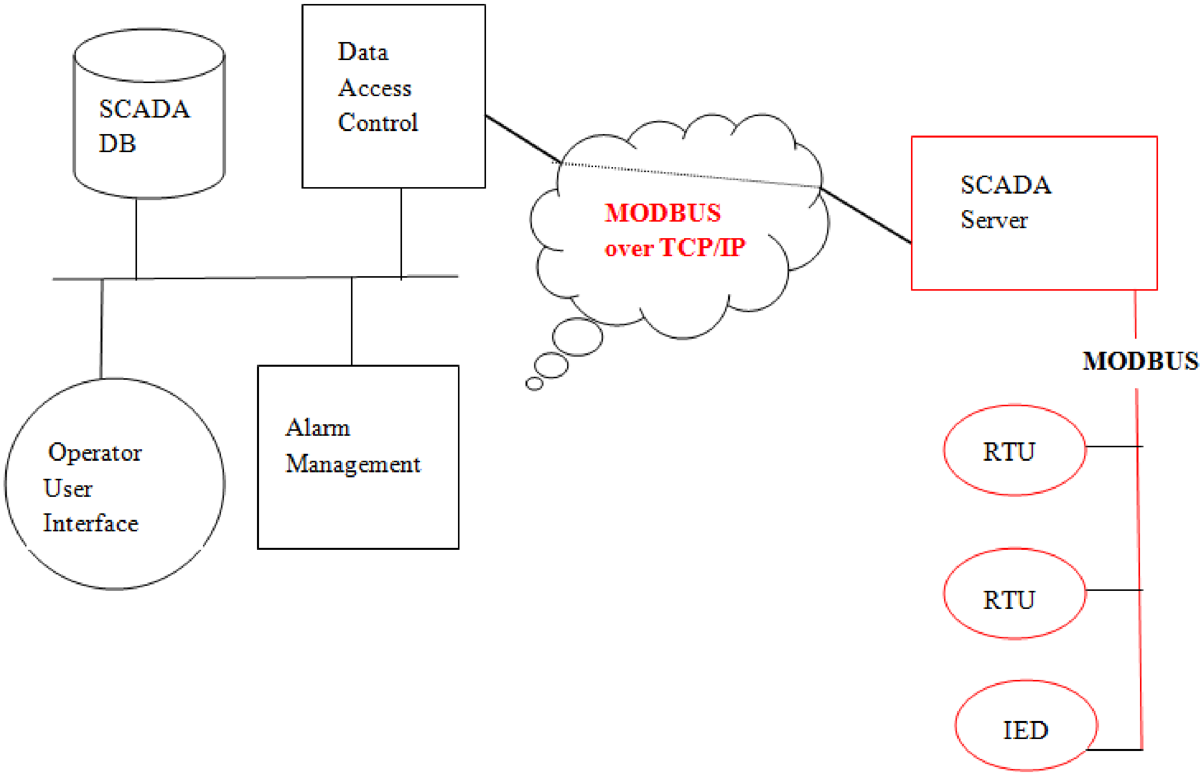


Figure 2 Major Components in Power System SCADA Communication

2.2 MODBUS Protocol

This messaging protocol has been the industrial standard for communication among automation and control devices since 1979. Originally it was designed for serial communication. Recently, it has been modified to run over Ethernet and is at the application layer (level 7) of the OSI model. It provides client/server communication between devices connected on different types of networks. It uses the master slave method at the message level but peer-peer method at network level. The master or the client initiates the transactions with the slave devices (or servers) by sending requests and the slaves respond to the requests by performing the requested action. MODBUS can be implemented in three different ways:

- As a serial protocol in RTU mode.
- As a serial protocol in ASCII format.
- Ethernet based MODBUS TCP.

ASCII mode: Each byte is sent as two ASCII characters. Each ASCII character contains one Hexadecimal character. The format for each byte is as follows: Every byte has 1 start bit and 7 data bits. If even or odd parity is used, 1 bit is reserved for parity and 1 bit is used as stop bit. If parity is not used, two bits are used as stop bits. The error checking is done using LRC (Longitudinal Redundancy Check) error check.

RTU mode: The coding system used is 8–bit binary for a message which corresponds to two 4-bit hexadecimal characters. This has 1 start bit, 8 data bits with 1 start bit and 8 data bits. The parity and stop bits are similar to those in ASCII mode and the error check used is Cyclical Redundancy Check (CRC).

TCP/IP over Ethernet: MODBUS has been ported over TCP/IP and embedded into the payload of a TCP packet. Port 502 is reserved for this purpose. Currently, this is the most widely used MODBUS implementation. The query from the master using the MODBUS protocol consists of the device address, a function code corresponding to the action requested, any data to be transmitted, and an error–checking field as shown in Figure 3.

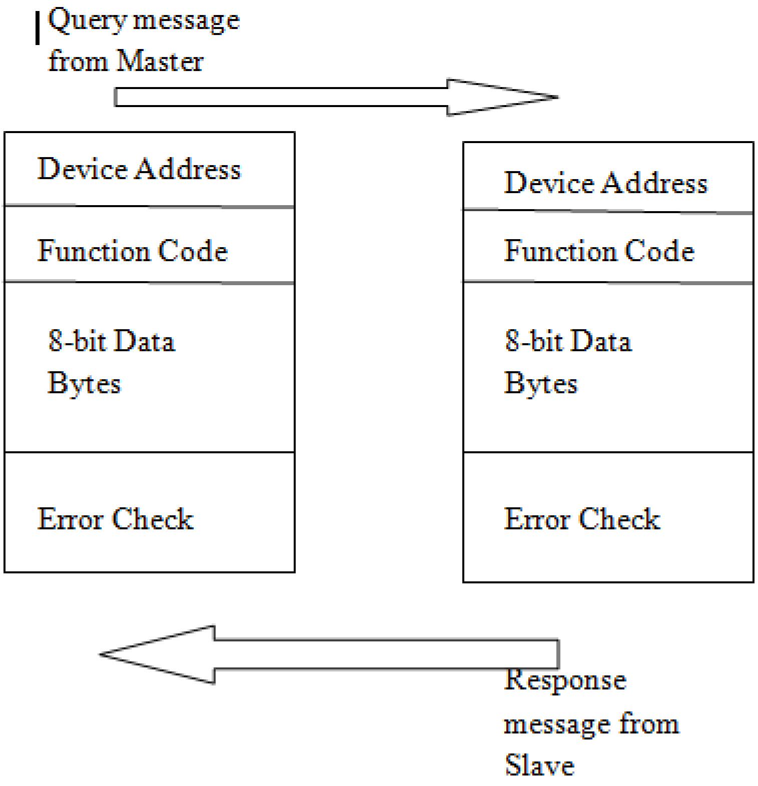


Figure 3 Master Slave Query Response Cycle

The data bytes contain additional information needed by the slave to perform the function indicated by the function code. The slave's response message constructed using MODBUS contains fields confirming the action taken, any data to be returned, and an error–checking field. If an error occurred which resulted in the slave device not performing the requested action, the slave sends an error message as the response. The case of error free MODBUS transaction is shown below in Figure 4:

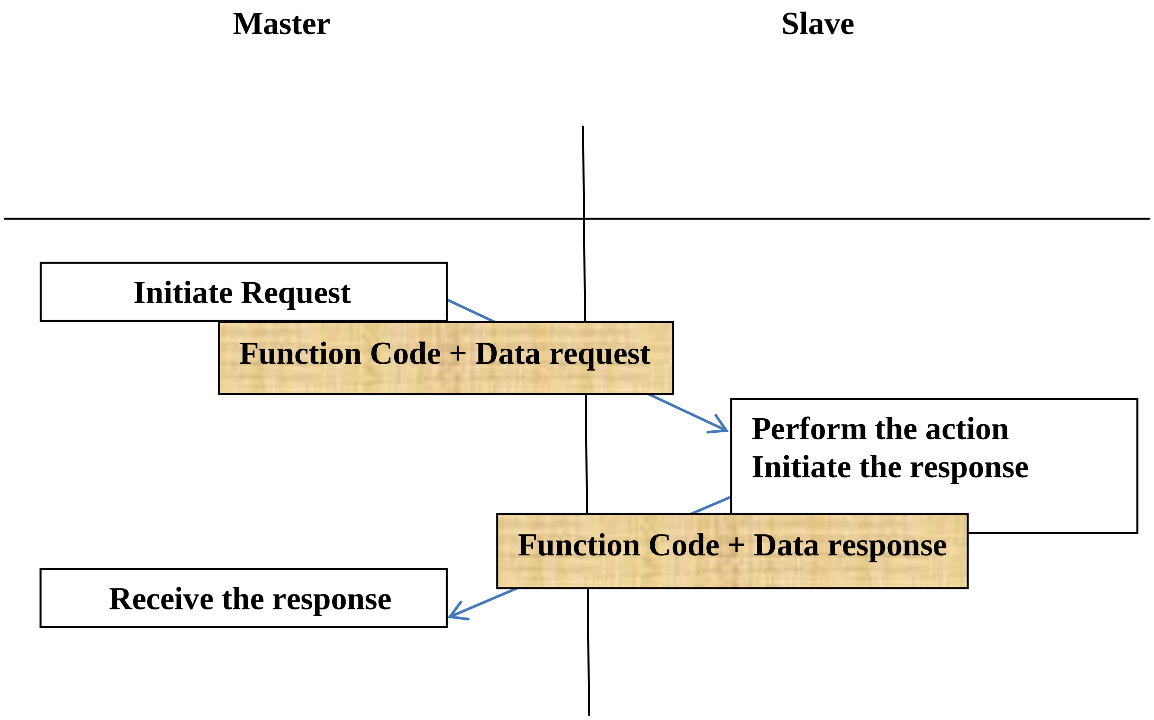


Figure 4 Error-free MODBUS Transaction

Apart from function code and data, there are additional fields in the MODBUS packet as shown in Figure 5.

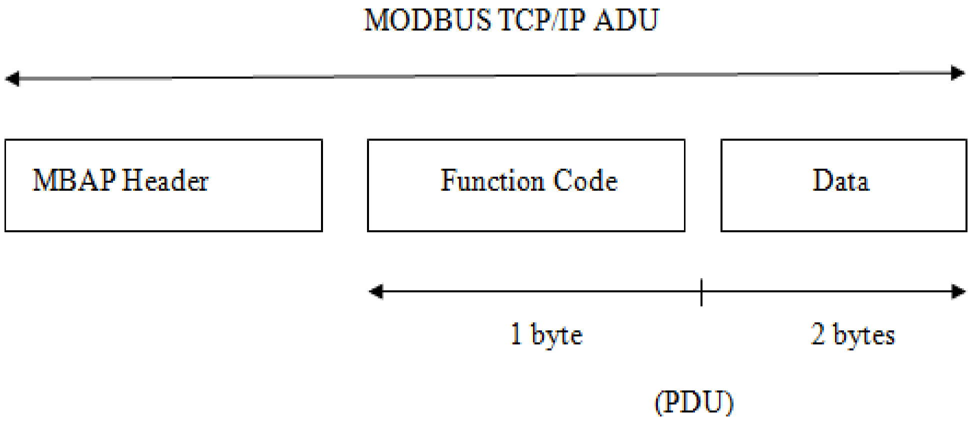


Figure 5 General Format of MODBUS Request/Response over TCP/IP

The function code together with the data called as a Protocol Data Unit (PDU) is independent of the underlying communication layers. Function code is of 1 byte. Hence the range is 1-255, in which 128-255 are reserved for exception responses. Data is of 2 bytes. The Protocol Data Unit together with certain additional fields constitute the ADU (Application Data Unit). These additional fields are called MBAP Header as shown in Figure 6 and explained in Table 1.

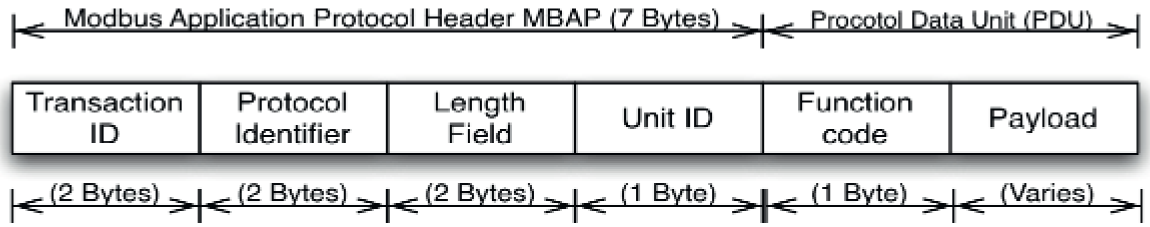


Figure 6 MBAP Header

Table 1 Fields in MODBUS ADU Header

Field Name	**Length(in bytes)**	**Description**
Transaction Identifier	2	Identifier for the MODBUS Request/Response transaction
Protocol Identifier	2	0 for MODBUS
Length	2	Byte count of the fields, Unit Identifier and Data fields
Unit Identifier	1	Identifier for any remote slave connected on a serial line or other buses

There are three categories of function codes used in MODBUS request/reply.

1. **Public function codes**: These are unique function codes which are publicly documented. Some examples include 0x01(Read Coils), 0x05(Write Single Coil), 0x04(Read Input register), 0x06(Write Single Register) etc. The list has been shown in the Appendix A.
2. **User- Defined function codes**: These include two ranges: 65-72 and 100-110. These codes can be selected and implemented by the user.
3. **Reserved Function Codes**: These are not available for public use and are used by some companies for legacy products.

The PDUs are classified into three different kinds as follows:

MODBUS Request PDU, mb_req_pdu = {function_code, request_data} where function_code corresponds to the 1-byte function code requested through the client API and request_data is the n-byte data containing references to the register address, offsets, variable counts etc.

MODBUS Response PDU, mb_resp_pdu = {function_code, response_data} where function_code is the same as the one in the Request PDU and response_data field is function code dependent.

MODBUS Exception Response PDU, mb_excep_rsp_pdu= {function_code, response_data} where function_code is the sum of MODBUS function code and 0x80. An example of a request and response PDU in MODBUS transactions is shown in Table 2.

Table 2 Example of MODBUS Request Response Transaction

Request PDU		Response PDU	
Function	01	Function	01
Starting Address Hi	00	Byte Count	03
Starting Address Lo	13	Outputs status 27-20	CD
Quantity of outputs Hi	00	Outputs status 35-28	6B
Quantity of outputs Lo	13	Outputs status 38-36	05

The request PDU corresponds to “read 19 coils starting from the address 19”. The individual fields are to be interpreted as follows: The MODBUS function code 0x01 in both request and response PDUs corresponds to ‘Read coils’. The starting address is 0013 in hexadecimal and the quantity of registers to be read is also 0013 in hexadecimal as shown in the request PDU. In the response, the byte count 3 indicates that the ensuing 3 bytes are the data requested. The data bytes are CD, 6B and 05 in hexadecimal. The status of the registers 27–20 is shown as CD hex, or binary 1100 1101. Output 27 is the MSB, and output 20 is the LSB. In the last data byte, the status of outputs 38-36 is shown as 05 hex, or binary 0000 0101. Output 38 is in the sixth bit position from the left, and output 36 is the LSB of this byte. The remaining 5 high order bits are zero filled. The MODBUS requests and responses discussed have a direct correlation with the underlying physical system. Hence spoofing of such requests and responses can be used as a means of intruding into the physical system.

2.3 Intrusion Detection for SCADA

Intrusion Detection is the process of identifying intrusions in a system. The intrusion detection methods are classified into two main categories: misuse-based and anomaly-based [6].

1. **Misuse-based:** This is based on constructing signatures from known attack patterns and vulnerabilities. The patterns of known attacks are specified as rule sets. There are some languages like SNORT, RUSSEL, P-BEST etc. developed for representing the rule sets. The advantages are higher detection rate and lower false alarm rates for known attacks. The limitation is that it cannot detect unknown or novel attacks.
2. **Anomaly-based:** This is based on capturing the normal behavior of a system using some features extracted from the system considered. The intuition behind this approach is that something that is abnormal can be suspicious. This usually involves two phases: an initial training phase and a detection phase. During the training phase, normal behavior of the system being monitored is learnt and a profile is built to represent this normal behavior. During the detection phase, the behavior of the system is observed and when it is deviant from the normal profile, an anomaly flag is raised. The advantage of this approach is that it can detect new attacks which misuse-based approaches cannot. The drawbacks are
    - High false alarm rates
    - False positives and negatives are highly dependent on training data.

- If the normal behavior of the system changes over time, normal profiles also need to be updated accordingly.

Following are few of the anomaly-based methods:

Statistical learning: In this, normal system behavior is captured using statistical models. For example, measures like mean, standard deviation are used to represent the normal range of certain parameters during training. During detection, the deviation from the normal range is used to assign an anomaly score for every parameter. The weighted sum of the anomaly scores obtained for various parameters is the final score indicative of the anomalous behavior of the system.

Specification based: The normal behavior is represented using sequential patterns. For example, a sequence of function calls or sequences of state changes are extracted from the system during training phase and a model corresponding to normal behavior is built. During detection, the same states/function calls extracted from the system are organized into sequences and compared against the normal model established during training phase. It has a high potential for generalization and even new types of attacks can be detected. The drawback is that it requires substantial efforts to characterize the model for normal behavior.

The suitability of these methods to SCADA traffic has been studied by various people as explained in the paper on SCADA specific Intrusion Detection Techniques [7]. For example, Steven et al. [8] have proposed a specification-based multi-IDS algorithm based on a formal model of the underlying MODBUS/TCP. They have used SNORT rules to analyze violations in communication patterns. Yang et al. [9] have proposed an anomaly detection method for SCADA systems based on features like network traffic, link utilization, CPU usage etc. Their method represents the features as a vector which is fed as input to an auto associative kernel regression (AAKR) model which predicts the correct versions of the input. Residuals are formed by comparing the observed values with the predictions from the model. A binary hypothesis technique called the sequential probability ratio test (SPRT) is applied to the residuals to determine if the residuals correspond to a normal or abnormal distribution. The attacks considered are DOS, ping flood, jolt2 etc. Oman and Phillips [10] have proposed a misuse-based intrusion detection approach and event monitoring by producing signatures for unauthorized access in SCADA devices. Details about each SCADA device like its IP address, telnet port, and legal commands for each device are expressed using XML. A Perl program parses this XML and creates SNORT signatures which are used in detecting attacks later. There are also commercial MODBUS filtering modules like Cisco's Net filter which is rule-based [11]. Chabukswar et al. [12] have simulated a simple SCADA system consisting of a plant and a controller and studied the effects of attacks like Distributed Denial of Service (DDOS) on the communication network between the plant and the controller. Carcano et al. [13] have developed a state based approach to represent the internal critical states of the SCADA system caused by malicious MODBUS commands embedded in a single packet. Giani et al. [14] have developed a SCADA test bed based on Simulink/ State flow from Math works and tested attacks like Denial of Service attack on sensors, Integrity attacks, Phishing attacks, etc. Mallouhi et al. [15] have developed a test bed to simulate a SCADA client, server and power grid. They have also proposed an Autonomic Software Protection System (ASPS) which has rules to analyze MODBUS requests/responses. This system selectively drops packets and protects the SCADA systems from flooding and Denial of Service (DOS) Attacks.

Most of these existing methods are based on the parameters extracted from TCP/IP traffic or other generic ETHERNET parameters. There are few methods specific to MODBUS which are rule based and depends on the MODBUS request or response within a single packet. Our proposed approach differs from these in the following ways:

- It addresses the problem of intrusion detection at application layer level by leveraging the parameters in the MODBUS payload embedded within the TCP packet.
- It takes into account the sequence of MODBUS packets and not a single packet.
- It takes into account the physical state of the power system to model the cyber-physical relationship in the power system.
- It focuses on the design of an intrusion detection system which is suitable for implementation in field devices.

3. PROBLEM FORMULATION AND PROPOSED SOLUTIONS

This Section discusses the problem formulation and proposed solutions for local intrusion detection in SCADA field devices for smart grid. The architecture considered, the threat model and the attack scenarios considered for intrusion detection are discussed in Sections 3.1 and 3.2. The two approaches considered for intrusion detection, viz, the hash table and Bloom Filter are described in detail and their pros and cons are compared in Section 3.3. In Section 3.4, the process of enhancing the intrusion detection using physical information is presented.

3.1 Problem Formulation

The problem is to detect intrusions locally in SCADA field devices for smart grid. An intrusion detection system needs to be developed for this purpose. Such a system should be able to detect the malicious commands sent by an attacker to the SCADA field devices like RTUs, PLCs to affect the underlying power system components. These commands may be sent either by compromising the HMI or by compromising the communication link between the SCADA server and the field devices. The system has to work at the application level by extracting certain parameters from the traffic flow between the SCADA master and the field devices.

The architecture considered for this problem is a typical SCADA master-slave configuration as shown in Figure 7. The main components are HMI at the operator workstation, SCADA master, field devices like RTUs, PLCs and the communication links between them. The communication between SCADA master and field devices are assumed to follow MODBUS over the TCP/IP protocol.

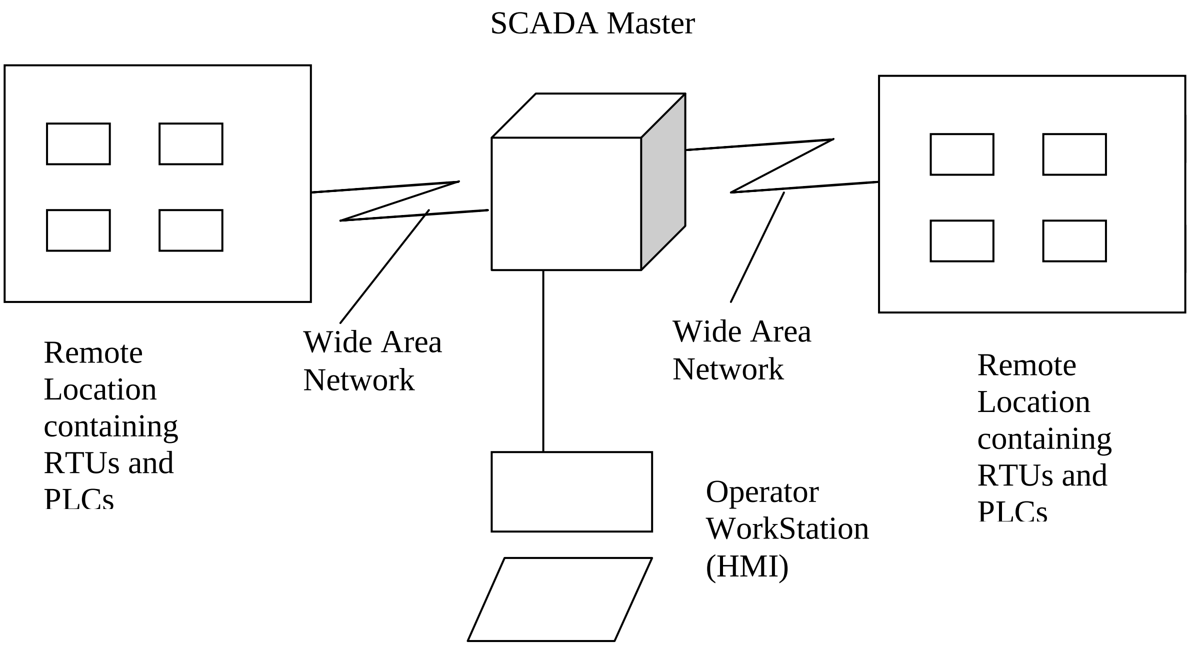

SCADA Master
Remote Location containing RTUs and PLCs
Wide Area Network
Wide Area Network
Remote Location containing RTUs and PLCs
Operator WorkStation (HMI)

Figure 7 SCADA Master-Slave Architecture

MODBUS protocol was chosen for the following reasons:

- It is the most widely used application layer protocol for communications between the SCADA master and field devices in industrial networks.
- MODBUS protocol has inherent vulnerabilities due to lack of authentication, authorization and encryption in Protocol Design and Specification and it is possible for an attacker to exploit these vulnerabilities easily.

Though the problem is formulated using MODBUS, the same can be applied to any SCADA protocol. The problem assumes a specific threat model which is detailed below.

3.2 Threat Model

The main threats considered are external targeted attacks. Threats from insiders are not considered. Also it is assumed that the attacker compromises a single specific component of the SCADA system like the HMI or communication link at any point of time. Colluded distributed attacks are not considered. The attack scenarios under this assumed threat model are as follows:

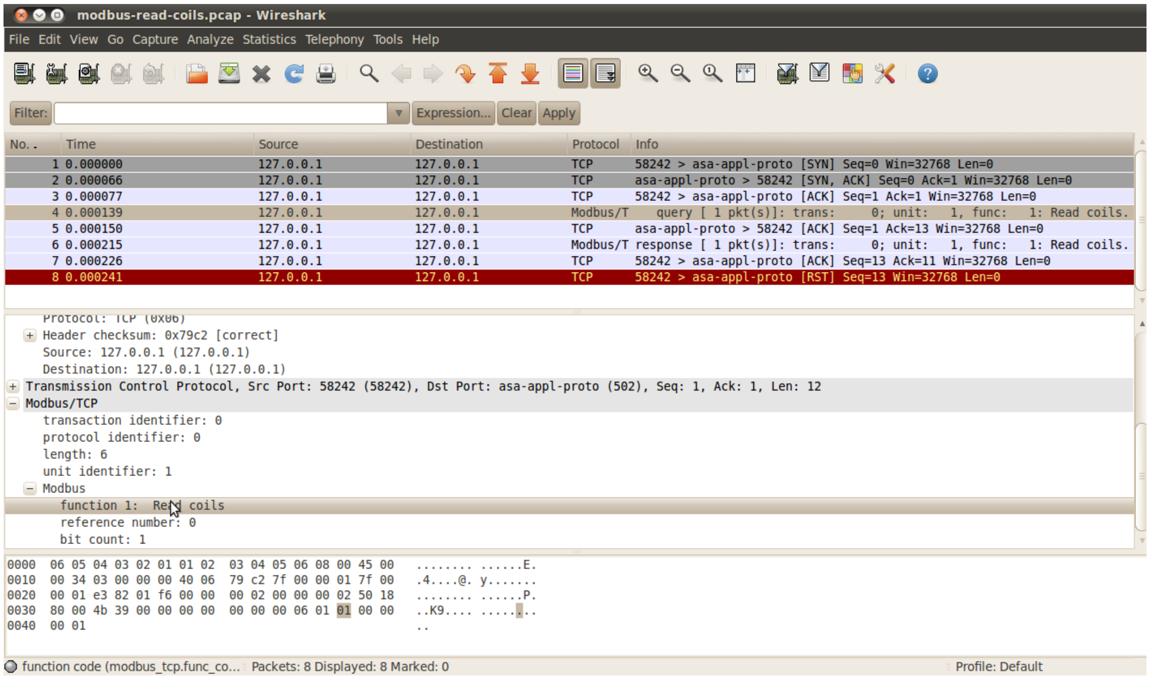
1. **HMI Compromise**: This is the case where the attacker has compromised the HMI and can send commands to destabilize the grid. Common HMI commands in a power system include read bus line status, read transformer status, read/change magnitude and phase angle of the bus, open/close circuit breakers etc. While the attacker can manipulate these commands as per his attack objectives, he can also make sure that the display at the operator interface does not show the actual state of the system.
2. **Man-in-the-middle attack**: This involves intercepting communication between SCADA server and field devices and sending spoofed packets to them. This can be used by the attacker as a means of performing illegal writes to coils or registers, restarting MODBUS server multiple times, etc. These operations can affect the states of components in the power system and can eventually destabilize the grid.

3.3 Proposed Solutions

Our proposed approach of intrusion detection is based on the intuition that the above mentioned attacks get manifested as anomalies in the MODBUS request/response traffic patterns. These can be detected by establishing a normal profile and by comparing against it. To establish the normal profile, parameters such as function code and data in MODBUS TCP/IP PDU are extracted. The entire process can be implemented using the following two steps.

3.3.1 Data Extraction Using n-gram Analysis

N-gram is a contiguous sequence of n items. Data extraction involves extracting the MODBUS PDU information (function code and data) from the traffic between the MODBUS server and the field devices using a sliding window of some length, say ‘n’. The size of ‘n’ is dependent on the nature of traffic. By varying the window size between 5 and 12, we find that the performance is better for a window size of 5.   A sample trace of the MODBUS payload within the TCP packet is shown in Figure 8.


modbus-read-coils.pcap - Wireshark
File Edit View Go Capture Analyze Statistics Telephony Tools Help
Filter:
Expression...
Clear
Apply

```
No. .  Time        Source      Destination  Protocol  Info
     1 0.000000    127.0.0.1   127.0.0.1    TCP       58242 > asa-appl-proto [SYN] Seq=0 Win=32768 Len=0
     2 0.000066    127.0.0.1   127.0.0.1    TCP       asa-appl-proto > 58242 [SYN, ACK] Seq=0 Ack=1 Win=32768 Len=0
     3 0.000077    127.0.0.1   127.0.0.1    TCP       58242 > asa-appl-proto [ACK] Seq=1 Ack=1 Win=32768 Len=0
     4 0.000139    127.0.0.1   127.0.0.1    Modbus/T     query [ 1 pkt(s)]: trans:     0; unit:   1, func:   1: Read coils.
     5 0.000150    127.0.0.1   127.0.0.1    TCP       asa-appl-proto > 58242 [ACK] Seq=1 Ack=13 Win=32768 Len=0
     6 0.000215    127.0.0.1   127.0.0.1    Modbus/T  response [ 1 pkt(s)]: trans:     0; unit:   1, func:   1: Read coils.
     7 0.000226    127.0.0.1   127.0.0.1    TCP       58242 > asa-appl-proto [ACK] Seq=13 Ack=11 Win=32768 Len=0
     8 0.000241    127.0.0.1   127.0.0.1    TCP       58242 > asa-appl-proto [RST] Seq=13 Win=32768 Len=0
```

```
    Protocol: TCP (0x06)
  + Header checksum: 0x79c2 [correct]
    Source: 127.0.0.1 (127.0.0.1)
    Destination: 127.0.0.1 (127.0.0.1)
+ Transmission Control Protocol, Src Port: 58242 (58242), Dst Port: asa-appl-proto (502), Seq: 1, Ack: 1, Len: 12
− Modbus/TCP
    transaction identifier: 0
    protocol identifier: 0
    length: 6
    unit identifier: 1
  − Modbus
      function 1:  Read coils
      reference number: 0
      bit count: 1
```

```
0000  06 05 04 03 02 01 01 02  03 04 05 06 08 00 45 00   ........ ......E.
0010  00 34 03 00 00 00 40 06  79 c2 7f 00 00 01 7f 00   .4....@. y.......
0020  00 01 e3 82 01 f6 00 00  00 02 00 00 00 02 50 18   ........ ......P.
0030  80 00 4b 39 00 00 00 00  00 00 00 06 01 01 00 00   ..K9.... ........
0040  00 01                                              ..
```

```
function code (modbus_tcp.func_co...   Packets: 8 Displayed: 8 Marked: 0                Profile: Default
```

Figure 8 Sample Trace for MODBUS TCP/IP Packet

We can see that the function code in the payload is ‘1’ which corresponds to ‘read coils’. Say, the MODBUS function codes in the five contiguous packets are 1, 5, 5, 1, 5 respectively, n-gram analysis to extract the function codes using a sliding window of length 5 will yield a sequence 15515. Such sequences are used in training and detection as explained below:

3.3.2 Training and Anomaly-Based Detection

The data for training is based on the various MODBUS register and command assignments discussed in [16]. The preliminary data is taken from [16] and some similar random data patterns are created to be used as the training data. The MODBUS function codes and the data like coil addresses from this training set are subject to n-gram analysis and then represented in a compressed form using two different approaches, hash table and Bloom Filter.

3.3.2.1 Hash Table

This is a data structure that is used to quickly locate a data record. Its construction can be explained as follows: Consider a set of input elements of size ‘n’. Each element in this set is called as a key. These keys are mapped to a one-dimensional array using a method called hashing. The functions which are used for hashing are called “hash functions”. Examples of such functions are shown in the Appendix A. These hash functions map the keys to hash values which correspond to indices in an array. Each index in the array is associated with a list of records which hashed to the corresponding index. This can be explained using an example as follows:

Let us consider a list of words A, C, THE, YES, PART to be stored in a hash table. Consider a simple hash function which adds the position of each alphabet in a word followed by modulo-13 operation. The result of such hashing will be 1%13, 3%13, 33%13, 49%13, 55%13 which is equal to 1, 3, 7, 10, 3 respectively. Hence the key to hash value mapping will be as shown in Table 3.

Table 3 Hash Key Value Mapping

A	1
C	3
THE	7
YES	10
PART	3

These hash values correspond to the indices in an array and each index points to a list of keys as shown in Figure 9.

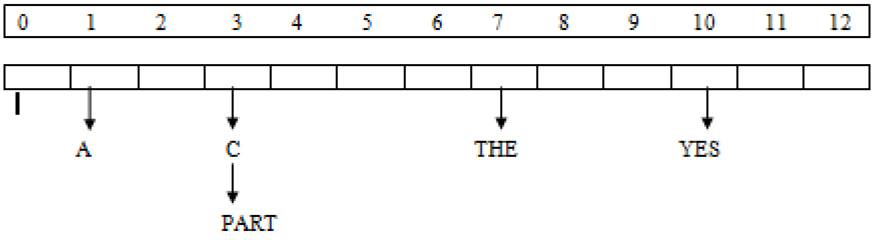


Figure 9 Hash Table Construction

Such a construction is called as hash table and can be used for storing the sequence of function codes and data obtained during n-gram analysis of the training data set. After the training data set is represented in the form of a hash table, this can be used for anomaly detection.

The n-grams obtained from communication patterns to be tested are hashed using the same hash function used in training. The output obtained after hashing corresponds to the array index where the record has to be searched for. The list at the particular index is traversed and if the input data is present in the list, the communication pattern is normal. If not, an anomaly score needs to be assigned. The efficiency of this method depends on the lengths of the linked lists which in turn depend on the hash function used and the size of the array.

The disadvantage in this approach is the overhead for sequential searching in the linked list. As an alternative to this approach, another data structure called Bloom Filter is considered for our implementation which is described below.

3.3.2.2 Bloom Filter

Bloom Filter is a probabilistic data structure which is a memory-efficient alternative to the traditional look up hash method. According to [17], Bloom filter can be used to represent a set in a compact form if the effect of false positives can be mitigated. It has two components; a set of ‘k’ hash functions and a bit vector of length ‘m’.

Consider a set S={x_1, x_2, x_n} of n elements given as input to the Bloom Filter. Consider the two components of the Bloom Filter as follows: a set of ‘k’ independent hash functions h_1, $h_{2\ \ldots}$ h_k with range {1, 2… m} and a vector ‘v’ of ‘m’ bits, all set to zero initially.

To insert an element s϶S into the vector ‘v’ during training, it is hashed ‘k’ times and the bits in ‘v’ at the indices corresponding to the output of the hash functions are set to 1. A particular bit may be set to 1 multiple times by different inputs. To test if an element ‘b’ exists in the set S, the bits at positions h_1(b), h_2(b)……,h_k(b) are checked and even if any one of them is zero, the element does not exist in the set S. Otherwise, the element may exist in the set. (i.e.) there is a probability for it to not exist as well.

In our approach, Bloom Filter is used to efficiently store the sequence of function codes and data obtained during n-gram analysis of the training data set. The n-grams are hashed ‘k’ times and stored in the Bloom Filter during training. The n-grams obtained from the live communication patterns during detection are also hashed using the same ’k’ hash functions used in training. The outputs, obtained after hashing, correspond to the indices in vector ‘v’ to be checked. If the bits at all of these indices are 1, the tested pattern may be valid. Even if a single bit is zero, the tested pattern does not exist in the training set.

3.3.2.3 Choice of Bloom Filter's Parameters

There is a trade –off between the memory requirements (m) of the Bloom filter and the false positive rate. The false positive rate can be tuned by the choice of parameters 'm' and 'k'. The relationship can be derived as follows:

After representing 'n' keys using the vector 'v' of size 'm', the probability that a bit in vector 'v' is still zero is given by $(1- 1/m)^{kn}$. Hence the probability of a false hit is given by

$$(1- (1- 1/m)^{kn})^{k} \sim= (1-e^{kn/m})^{k}$$

The right hand is minimized when **k= ln (2) * (m/n)** in which case it becomes

$$(1/2)^{k}= (0.6185)^{m/n}$$

Another criterion to reduce false positive rate is to have at least half of the bits set in the bloom array, according to [17].

The criteria for the selection of hash functions used in Bloom filter is as follows:

- The hash function must be independent and uniformly distributed so that the output is equally distributed over the hash space. A hash function which satisfies avalanche criterion generates a uniform distribution. Whenever a single input bit is complemented, if each of the output bits change with a probability of 0.5 for any combination of the remaining input bits, the corresponding hash function is said to satisfy avalanche criterion.

- It should be light weight unlike some cryptographic hash functions. Examples of fast, simple hashes used in Bloom filters include murmur, FNV etc.
- It should produce minimal collisions. A collision occurs when two inputs in S hash onto the same index in the vector ‘v’. If the number of elements (keys) to be hashed exceeds the size of the vector, then no hashing algorithm can hash every key to a unique index. Hence it is impossible to make any hash function collision free. It is only possible to find a hash function that produces minimal collisions for a particular considered set of data.

Various hash functions have been proposed and implemented in literature. Few such hash functions are as follows:

1. **Shift-Add-XOR hash**: This is mainly designed for hashing strings. It uses a combination of left and right shifts for mixing. The amount of shifts is usually prime. An example is given in the Appendix A. This is found to be simple but fails avalanche test and hence not suitable for implementing Bloom Filter.
2. **FNV hash**: FNV hashing provides a high dispersion which is suitable for nearly identical strings such as URLs, hostnames, filenames, etc. The core of the FNV-1 hash algorithm [18] is given in the Appendix A. FNV hashes are fast and has a low collision rate but they do not satisfy the property of setting at least half of the bits in the Bloom Filter.
3. **One-at-a-Time hash**: A simple algorithm for this hash function is given in the Appendix A. This has good avalanche behavior but do not satisfy the property of setting at least half of the bits in the Bloom Filter.

4. **Jenkins Hash**: This has good avalanche behavior and has been well tested for its suitability for various types of input but the algorithm is too complicated and not suitable for light-weight applications.

5. **Murmur Hash**: The name comes from the following sequence of operations which mixes the bits of a value. k *= m; x = rotate_left (k,r); (multiply and rotate). When this is repeated many times, k ends up being pseudo-randomized. The choice of the two mixing constants, m and r decide the collision resistance of this hashing function. This hashing function is found to be suitable for Bloom Filters due to its excellent avalanche behavior, good performance and collision resistance.

Hence the hash function chosen for our implementation is murmur hash which takes the n-grams extracted before as input and hashes them. These hash values are used to the set the bits and perform look ups during training and detection respectively.

3.3.2.4 Comparing Hash Table and Bloom Filter

Hash Table and Bloom Filter are compared based on the following criteria:

1. **Space and Time complexity :** Look-ups in STL container classes used for implementing hash tables are of complexity O (log n) where as both lookups and insertion are of order O (1) in Bloom Filters. Hence time complexity of Bloom Filters is lesser than that of hash tables. Space complexity is also higher in STL containers as every entry contains a hash value and a pointer to a list of elements where as Bloom Filter contains a single array with zeros and ones.

2. **Ability to dynamically update the data structure as more data becomes available:** Updating the data structure involves hashing and adding an element to the list in Hash table. It involves hashing and flipping a bit in an array in Bloom Filter. Hence insertions are slightly more involved in hash table though there are no considerable differences between the two.
3. **False positives and false negatives:** Bloom Filter can determine if an element may exist in the set or not where as Hash table determines if an element does exist in the set or not. Also many inputs can set the same bit in Bloom Filter and there is no way to perform a reverse mapping. Hence false positives are theoretically more in Bloom Filter compared to hash table. But one of the advantages of Bloom Filters is that false positive rates can be adjusted by the choice of hash function used and other Bloom Filter parameters. Also there is no possibility of having false negatives (returning false for lookups when a value actually exists in the set) by the very nature of its construction.

Considering the pros and cons of both approaches, Bloom filter is chosen to test the intrusion detection for MODBUS traffic due to the following reasons:

- Extremely compact way to represent a set and hence can be suitable for representing the regular traffic patterns in SCADA communication.
- Space-efficient compared to other data structures like arrays, linked lists and hash tables and used in many network applications [17]. Hence this can be suitable for light-weight implementation at resource constrained devices like PLCs, RTUs.

- False negatives do not occur and there is a trade-off between the false positive rate and memory requirements.

3.4 Including Physical Information for Intrusion Detection

The discussed intrusion detection approach takes into account only the communication patterns in the SCADA networks. However as these patterns are strongly correlated with the physical information in the power system, it is good to consider the possibility of including physical states in the intrusion detection. This can be explained with respect to the operating states of a power system as shown in Figure 10:

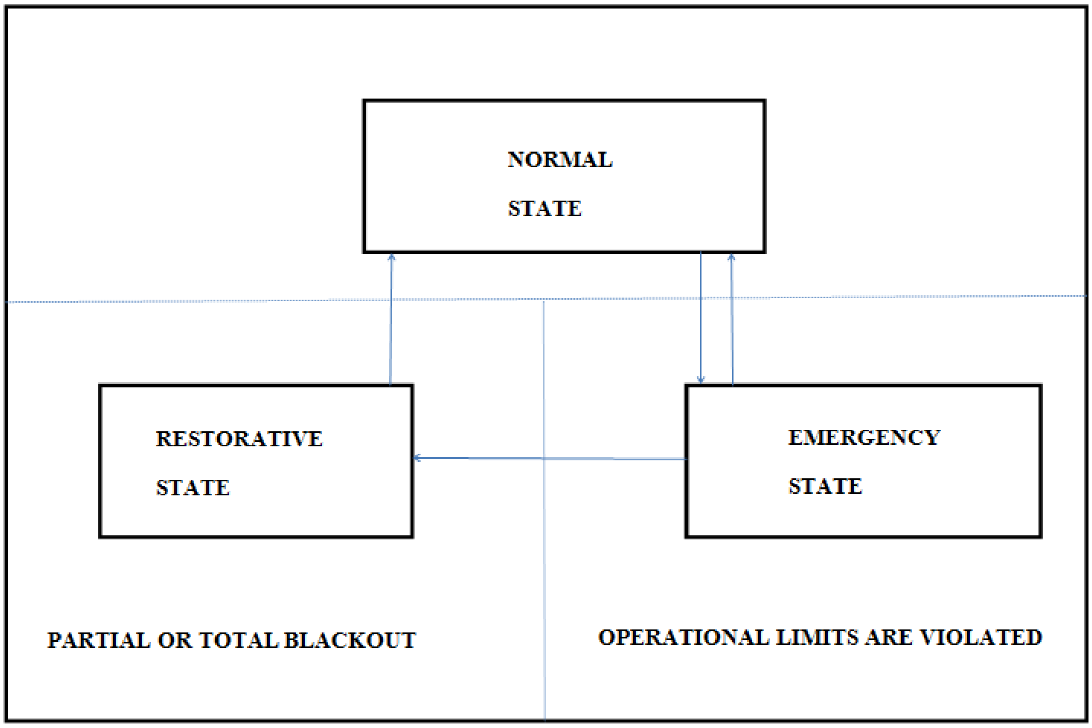


Figure 10 State Diagram for Power System Operation

In the normal operating state, the total load power can be supplied by existing generators without violation of any operational constraints. Examples of some operational constraints are limits on transmission line flows, voltage magnitudes, etc. In the emergency operating state, though some operational constraints are violated, the power system still supplies power to all the loads. In this state, the system operator takes corrective measures to bring the system back to normal state. The corrective measures can include disconnecting few loads, lines, opening and closing of breakers etc. If these measures fix the operating limit violations and makes the system stable with a reconfigured topology, some load versus generation balance has to be restored for the system to start supplying all loads again. This state is known as restorative state.

Hence the set of actions performed by the system operator vary according to the operating states of a power system. This in turn affects the way in which normal and abnormal behavior can be defined for the MODBUS request/response functions. For example, the set of MODBUS commands which may be anomalous under normal operating state may be normal under emergency operating conditions, as the operator has to take certain corrective measures under emergency state.

Hence the training data should be different for normal, emergency and restorative states and while the intrusion detection system is used on line, the detection should take into account the current state of the power system. If the physical state of the system is not considered, there are more possibilities for normal MODBUS traffic to be classified as abnormal and vice versa. For example, the function code 0x01 in MODBUS corresponds to “Read Coils” operation. This command also restarts the device and runs some start-up tests. Seeing this function code frequently within a time window may be abnormal under normal system state. However, this can be a valid operation under emergency state. If the intrusion detection system does not take this physical state into account while assigning an anomaly score to the tested communication pattern, normal traffic would be assigned a higher anomaly score. This results in higher false positives. This shows the necessity to include physical state of the power system into consideration during intrusion detection and this requires a constant feedback from the power system to the field devices like RTUs, PLCs.

The physical state of the power system is usually estimated using a process called state estimation. According to [19], if a power system has 'l' measurement devices and 'r' state variables, an adversary needs to compromise at least l-r+1 measurements to ensure that he can falsify state estimation. However l-r+1 is usually large for a power system. Even an IEEE 300-bus test system has l-r+1 =824. This shows the difficulty level for the attacker to falsify the physical state of the power system and in turn the robustness of our intrusion detection approach by including the physical information.

4. SIMULATIONS

The overall setup, tools and software and the data sets used in the implementation of the proposed Bloom Filter approach are discussed in Section 4.1. The results obtained are discussed in Section 4.2.

4.1 Setup and Tools Used

The overall setup used in the training and detection phases of the proposed Bloom Filter approach is explained as follows: The major tools used are Wireshark, a network protocol analyzer and C++. The data extraction, creation of Bloom Filter during training and anomaly detection are implemented in software using C++. The training phase is represented in Figure 11.

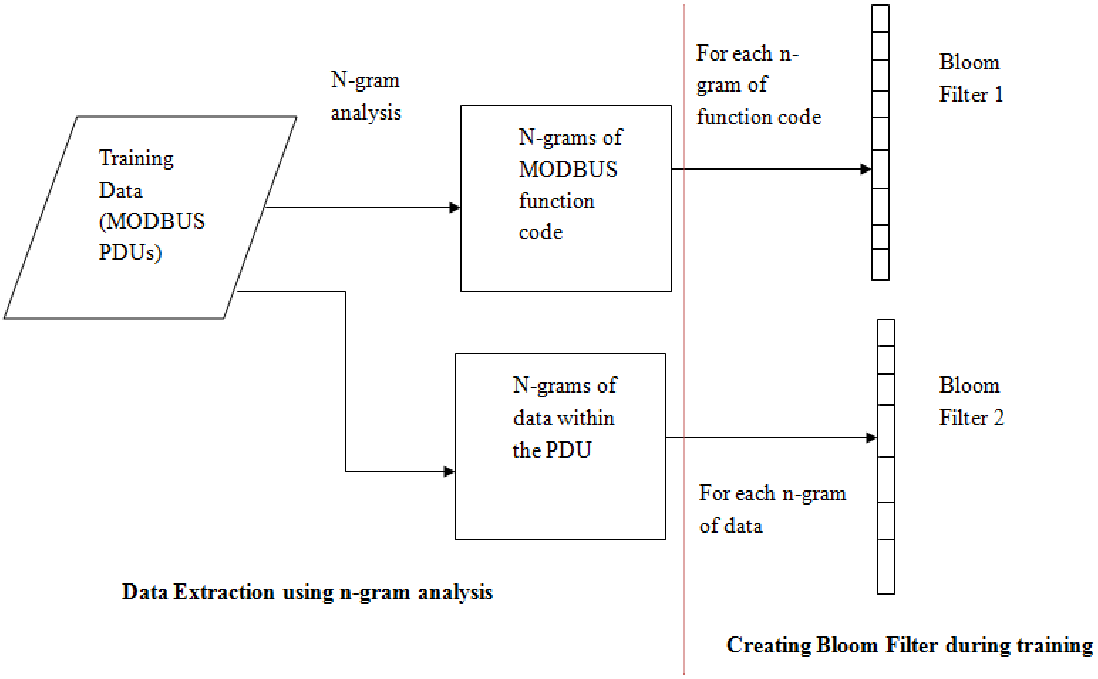
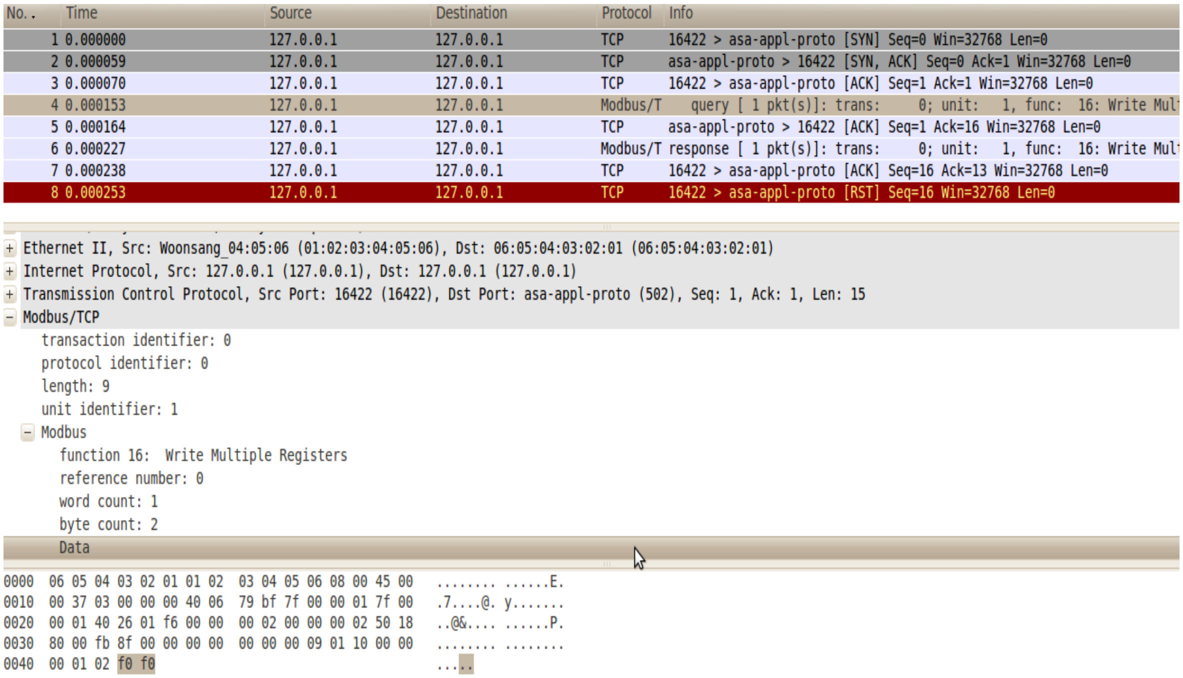

N-gram analysis
Training Data (MODBUS PDUs)
N-grams of MODBUS function code
For each n-gram of function code
Bloom Filter 1
N-grams of data within the PDU
For each n-gram of data
Bloom Filter 2
Data Extraction using n-gram analysis
Creating Bloom Filter during training

Figure 11 Overall Setup for Training Using Bloom Filter

The training data is taken from the MODBUS manuals [16]. The PDUs from these have been analyzed using Wireshark. A sample Wireshark trace of a MODBUS PDU is shown in Figure 12.



```
No..  Time        Source      Destination  Protocol  Info
   1 0.000000     127.0.0.1   127.0.0.1    TCP       16422 > asa-appl-proto [SYN] Seq=0 Win=32768 Len=0
   2 0.000059     127.0.0.1   127.0.0.1    TCP       asa-appl-proto > 16422 [SYN, ACK] Seq=0 Ack=1 Win=32768 Len=0
   3 0.000070     127.0.0.1   127.0.0.1    TCP       16422 > asa-appl-proto [ACK] Seq=1 Ack=1 Win=32768 Len=0
   4 0.000153     127.0.0.1   127.0.0.1    Modbus/T    query [ 1 pkt(s)]: trans:     0; unit:   1, func:  16: Write Mul
   5 0.000164     127.0.0.1   127.0.0.1    TCP       asa-appl-proto > 16422 [ACK] Seq=1 Ack=16 Win=32768 Len=0
   6 0.000227     127.0.0.1   127.0.0.1    Modbus/T response [ 1 pkt(s)]: trans:     0; unit:   1, func:  16: Write Mul
   7 0.000238     127.0.0.1   127.0.0.1    TCP       16422 > asa-appl-proto [ACK] Seq=16 Ack=13 Win=32768 Len=0
   8 0.000253     127.0.0.1   127.0.0.1    TCP       16422 > asa-appl-proto [RST] Seq=16 Win=32768 Len=0

+ Ethernet II, Src: Woonsang_04:05:06 (01:02:03:04:05:06), Dst: 06:05:04:03:02:01 (06:05:04:03:02:01)
+ Internet Protocol, Src: 127.0.0.1 (127.0.0.1), Dst: 127.0.0.1 (127.0.0.1)
+ Transmission Control Protocol, Src Port: 16422 (16422), Dst Port: asa-appl-proto (502), Seq: 1, Ack: 1, Len: 15
- Modbus/TCP
    transaction identifier: 0
    protocol identifier: 0
    length: 9
    unit identifier: 1
  - Modbus
      function 16:  Write Multiple Registers
      reference number: 0
      word count: 1
      byte count: 2
      Data

0000  06 05 04 03 02 01 01 02  03 04 05 06 08 00 45 00   ........ ......E.
0010  00 37 03 00 00 00 40 06  79 bf 7f 00 00 01 7f 00   .7....@. y.......
0020  00 01 40 26 01 f6 00 00  00 02 00 00 00 02 50 18   ..@&.... ......P.
0030  80 00 fb 8f 00 00 00 00  00 00 00 09 01 10 00 00   ........ ........
0040  00 01 02 f0 f0                                     .....
```

Figure 12 MODBUS Packet in Wireshark

The packet shown consists of the function code 16 (0x10) and data 0xf0f0. During training phase, such information is extracted from the MODBUS PDUs and few more similar patterns are generated using random number generators. These training data are stored in a file and then function codes and data are extracted from this file using n-gram analysis. Then they are stored in two different files as shown in Figure 11. Each n-gram is then hashed and stored in a compressed form using Bloom Filter. Two different Bloom filters are created, one for function code and the other for the corresponding data which will be used in the detection later. The setup used for the detection is shown in Figure 13.

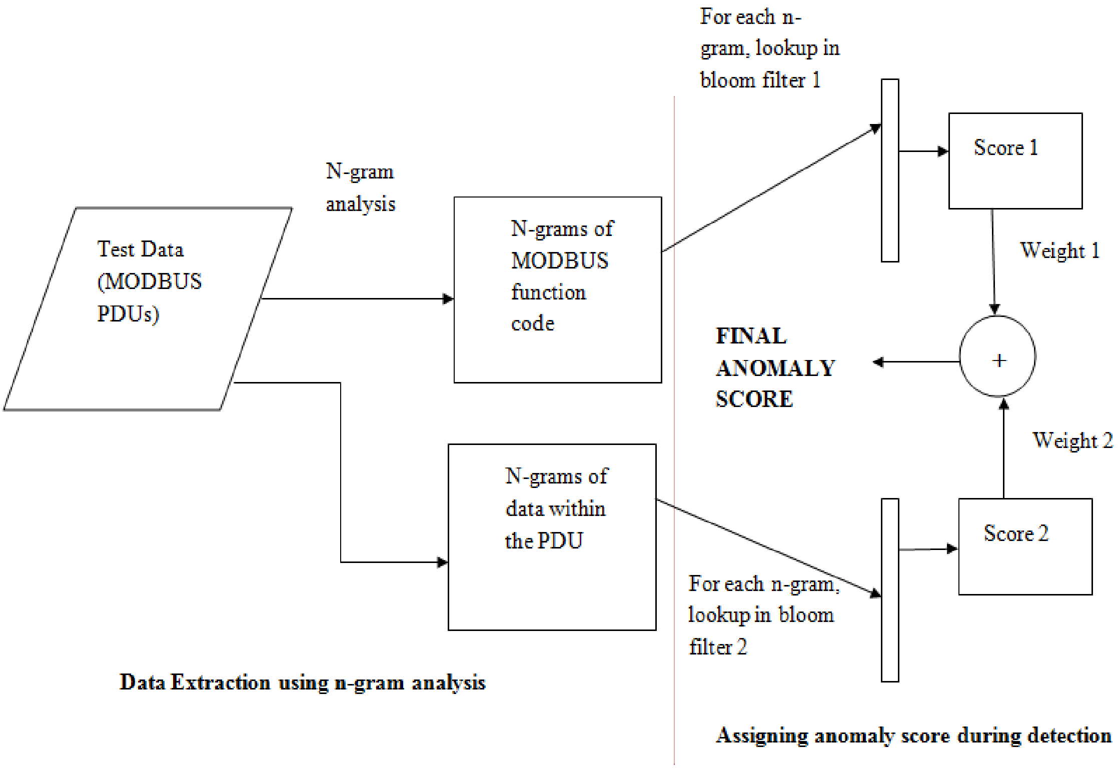

For each n-gram, lookup in bloom filter 1
N-gram analysis
Test Data (MODBUS PDUs)
N-grams of MODBUS function code
Score 1
Weight 1
FINAL ANOMALY SCORE
+
Weight 2
N-grams of data within the PDU
Score 2
For each n-gram, lookup in bloom filter 2
Data Extraction using n-gram analysis
Assigning anomaly score during detection

Figure 13 Overall Setup for Anomaly Detection Using Bloom Filter

During the detection phase, different MODBUS PDUs corresponding to normal data and cyber attacked data are taken from [16] and using n-gram analysis, function codes and data are extracted as in the training phase. After data extraction, look-ups are performed for each n-gram against the Bloom filters created during training. Each bloom filter's result is used for assigning a score to the tested pattern. The weighted sum of the scores from the two filters is the final score signifying the confidence in the anomalous behavior of the tested pattern. Using this training and detection procedure, several experiments have been performed to test the intrusion detection approach.

4.2 Results and Discussion

From a training input set consisting of 100 MODBUS PDUs, information such as function code and data was extracted using a sliding window of length ‘5’ during training. The parameter of the Bloom filter ‘k’ (number of hash functions) was varied from 2 to 6 and the results of the intrusion detection system are tabulated in Table 4:

Table 4 Performance of the Intrusion Detection Approach with Varying ‘k’

Value of k	**2**	**4**	**6**
m(size of the Bloom filter)	289	580	870
No. of bits set in ‘v’ after training	202	404	606
Percentage of times normal pattern is classified as anomalous (false positive)	0	0	0
Percentage of times anomalous is classified as normal(false negative)	6%	2%	2%

In Table 4, the parameter ‘m’ was calculated using the formula

k= (ln 2) *(m/n). For example, when k=4, m= (n*k)/ (ln 2) = (100*2)/ (ln 2) = 580 which is the size of the bloom filter. As seen, the size of the bloom filter increases with the number of hash functions. Also, it can be seen that more than half of the bits are set in the vector ‘v’ in all three cases. This is one of the criteria to reduce the false positive rate as per [16].

By the nature of construction of bloom filter, false negatives are not possible where as false positive rate can be adjusted by tuning its parameters. However, Table 4 shows a constant false positive rate of 0 and a non-zero value for false negatives. This is because the false negative of the bloom filter corresponds to false positive of the overall system. This can be explained as follows: false negative in Bloom filter is equivalent to saying “a particular sequence is not valid when it is actually valid”. This is not possible because if the sequence is valid and is included in the training data set, it would be represented in a compressed form in the bloom filter. Hence any normal pattern represented by the filter during training will not be classified as anomalous during detection. In other words, the system cannot have false positives with proper training data set. The results are in accordance with this fact.

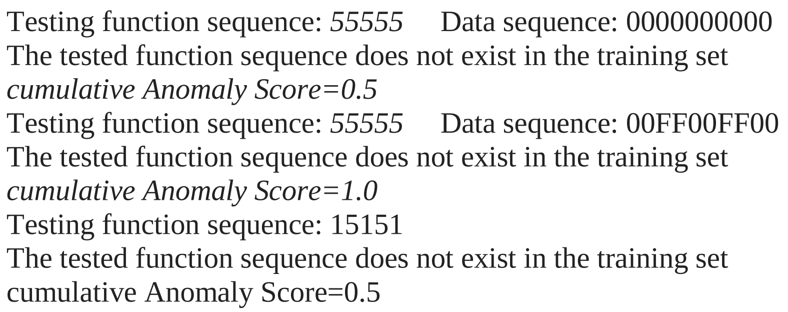
In a similar manner, the false positive rate of the bloom filter corresponds to the false negative rate of the overall system. Hence we see a non zero false negative rate for the system. We can also infer that there is a decrease in the false negative rate when ‘k’ is increased from 2 to 4. But when it is increased beyond 4 there is not much improvement in false negative rate. Hence it is sufficient to have 4 independent hash functions to achieve nominal detection for the chosen data. It is possible to generate these 4 independent and uniformly distributed hash functions by just varying the seed parameter in murmur hash.

Another test case is performed by varying n from 100 to 500 for a constant k=4 and the results of the intrusion detection system are shown in Table 5.

Table 5 Performance of the Intrusion Detection Approach with Varying 'n'

Value of n	**100**	**200**	**500**
m (size of the Bloom filter)	580	1159	2899
No. of bits set in 'v' after training	402	799	2000
Percentage of times normal pattern is classified as anomalous (false positive)	0	0	0
Percentage of times anomalous is classified as normal(false negative)	2.4%	0%	0%

From Table 5, we can infer that there is a decrease in the false negative rate when 'n' is increased from 100 to 200. But when it is increased beyond 200 there is not much improvement in false negative rate. Hence a training data set of size 200 (n=200) and 4 different hash functions (k=4) were chosen and the intrusion detection algorithm was tested. The sample output obtained from the detection algorithm is shown in Figure 14.



```
Testing function sequence: 55555     Data sequence: 0000000000
The tested function sequence does not exist in the training set
cumulative Anomaly Score=0.5
Testing function sequence: 55555     Data sequence: 00FF00FF00
The tested function sequence does not exist in the training set
cumulative Anomaly Score=1.0
Testing function sequence: 15151
The tested function sequence does not exist in the training set
cumulative Anomaly Score=0.5
```

Figure 14 Sample Output 1

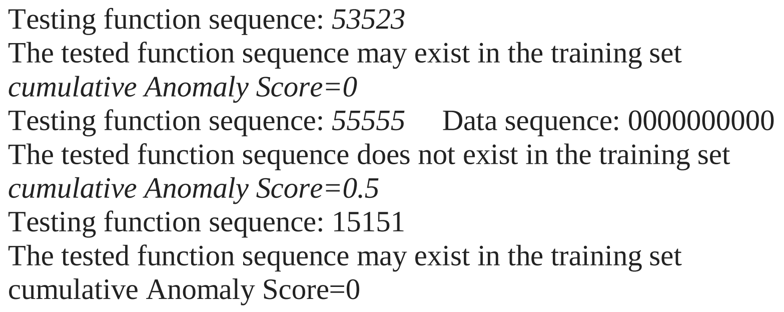
We can see that the anomaly score for the same function sequence ‘55555’ varies between 0.5 and 1.0 for different payload data. This can be explained as follows: Function code ‘5’ corresponds to “Force Single Coil” (Appendix A). This command forces a coil to be ON or OFF. This is determined using the data field. ‘FF’ in the data field corresponds to ON and ‘00’ corresponds to OFF. The function code sequence 15151 corresponds to “Read Coil- Write Coil- Read Coil-Write Coil-Read Coil”. As the command “read coil” restarts the device and runs some start-up tests, occurrence of this pattern quite frequently within a time window is classified as anomalous. The results of the intrusion detection algorithm are summarized in Table 6.

Table 6 Results for Training Using Data under “Normal” Conditions

Size of the training data set (n)	**200**
No. of hash functions used in Bloom Filter (k)	4
No. of bits in the vector (v) in Bloom Filter (m)	1159
No. of n-gram patterns used for testing	500
Percentage of times normal pattern is classified as anomalous (false positive)	0
Percentage of times anomalous is classified as normal(false negative)	2.4
Anomaly score for testing the function code sequence 55555 and data 00FF00FF00 (this alternates coil ON and coil OFF operations) –Data under a possible attack	1.0
Anomaly score for testing the function code sequence 55555 and data 0000000000 - Anomalous data, may not signify an attack and hence reduced anomaly score	0.5
Anomaly score for testing the function code sequence 15151 (this is valid data under emergency conditions)	0.5

From Table 6, we see that false negative rate is 2.4 %, false positive rate is zero and more than half of the bits are set in the vector ‘v’. This is similar to the results obtained by varying the parameters ‘k’ and ‘n’.

Another thing to be noted is that the pattern 15151 which is given an anomaly score of 0.5 in the above case may be valid under some conditions. For example, this may be valid under emergency state of the power system as some restorative actions are taken during this state. Hence training is carried out using the data under emergency conditions and sample output obtained is shown in Figure 15.



```
Testing function sequence: 53523
The tested function sequence may exist in the training set
cumulative Anomaly Score=0
Testing function sequence: 55555     Data sequence: 0000000000
The tested function sequence does not exist in the training set
cumulative Anomaly Score=0.5
Testing function sequence: 15151
The tested function sequence may exist in the training set
cumulative Anomaly Score=0
```

Figure 15 Sample Output 2

In this case, the same pattern 15151 is assigned an anomaly score of 0.

The results of the intrusion detection algorithm after training using emergency data are summarized in Table 7.

Table 7 Results for Training Using Data under "Emergency" Conditions

Size of the training data set (n)	**200**
No. of hash functions used in Bloom Filter (k)	4
No. of bits in the vector (v) in Bloom Filter (m)	1159
No. of n-gram patterns used for testing	500
Percentage of times normal pattern is classified as anomalous (false positive)	0
Percentage of times anomalous is classified as normal(false negative)	2.4
Anomaly score for testing the function code sequence 55555 and data 0000FF000000FF000000 (this alternates coil ON and coil OFF operations) – Data under a possible attack	1.0
Anomaly score for testing the function code sequence 55555 and data 00000000000000000000 (this alternates coil ON and coil OFF operations) - Anomalous data, may not signify an attack and hence reduced anomaly score	0.5

Table 7 Continued

Size of the training data set (n)	**200**
Anomaly score for testing the function code sequence 15151 (this is valid data under emergency conditions)	0

When training was done using normal data, the sequence 15151 was given an anomaly score of 0.5 and when trained using emergency data, the same sequence was given an anomaly score of 0. Hence without the knowledge of physical state of the system, it is not possible to ascertain the correct anomaly score. When information regarding the physical state of the power system is available, the anomaly scores are correlated with the physical information and the results are shown in Figure 16.

```
If the current state of the power system is "normal", then the results
become
Testing function sequence: 15151
The tested function sequence does not exist in the training set
State=normal
cumulative Anomaly Score=1
If the current state of the power system is "emergency", then the results
become
Testing function sequence: 15151
The tested function sequence does not exist in the training set
state=emergency
cumulative Anomaly Score=0
```

Figure 16 Sample Output 3

During the execution of this algorithm, input data sets of size 100, 1000 or greater than 1000 had an execution time of the order of few milliseconds. This shows that the time complexity of the algorithm is almost constant for various training input sizes. This is one of the advantages of Bloom Filter approach.

5. CONCLUSIONS

A Bloom Filter based intrusion detection approach for resource constrained SCADA field devices like PLCs, RTUs, etc. is proposed. This has been tested on SCADA traffic using MODBUS protocol. Bloom Filter has been compared with other data structures like hash tables, linked lists, etc and found to be more advantageous due to the following reasons:

- Low memory requirements.
- False positive rates for the overall intrusion detection system are zero for all the test cases performed.
- Unlike other data structures, Bloom filter does not store the information directly and hence it provides data anonymity.

The proposed approach has been tested with and without the knowledge of the current state of the power system. It is inferred that knowledge of the current state of the power system helps to reduce false positives in the proposed approach. This shows the importance of correlation between the information from SCADA components and information from the physical system to have better intrusion detection.

In most cases, SCADA server is protected by using a Demilitarized zone (DMZ) which acts as a buffer between SCADA and corporate networks. However our proposed system adopts a distributed intrusion detection approach which is meant to reside at end field devices like PLCs, RTUs, etc. The advantage is that the system can be partially operational even if few field devices are compromised.

5.1 Future Work

Following are few of the considerations for future work.

- As actual SCADA field data for power systems were not available, training and testing was mainly based on [16] and hence limited in nature. Testing the approach using actual field data should be considered for future work.
- The parameters used in our approach are function code sequences and the data in the MODBUS PDUs. More parameters can be added to the intrusion detection algorithm to make it more robust.
- While determining the overall anomaly score, the current algorithm assigns equal weight (=1) to all the parameters. This can be changed according to the importance of the parameters when more features are added to the intrusion detection system.
- The proposed approach has mainly focused on Man-in-the-middle and HMI compromise attacks. It has to be tested for attack scenarios.
- Alerts generated by the proposed intrusion detection system need to be correlated and aggregated for better reporting.
- The anomaly score obtained from the proposed approach can be correlated in a better manner with the power system parameters.
- The current algorithm has been implemented and tested in a standalone C++ environment. This can be integrated in a test bed containing SCADA server, HMI and power system components to study the interaction between the various systems.

REFERENCES

[1] D. Kundur, X. Feng, S. Mashayekh, S. Liu, T. Zourntos and K.L. Butler-Purry, “Towards modelling the impact of cyber attacks on a smart grid”, International Journal of Security and Networks, vol. 6, no. 1, pp. 2-13, Dec 2011.

[2] T. Dutzik and R. Sargent, After the Blackout Achieving a Cleaner More Reliable Electric System, September 2003[Online]. Available: https://pincdn.s3.amazonaws.com/assets/5fbd4cadc578fb62d4f872e8159f5ed2/US-After-the-Blackout.pdf Accessed 12/30/2011.

[3] R. Tsang, Cyber Threats Vulnerabilities and Attacks on SCADA Networks., [Online]. Available:http://gspp.berkeley.edu/iths/Tsang_SCADA%20Attacks.pdf Accessed 12/30/2011.

[4] N. Falliere, L.O. Murchu and E. Chie, W32.Stuxnet Dossier., February 2011[Online]. Available: http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf Accessed 12/30/2011.

[5] M. Labs, Protecting Your Critical Assets., December 2009[Online]. Available:http://www.mcafee.com/us/resources/white-papers/wp-protecting-critical-assets.pdf Accessed 12/30/2011.

[6] P. Ning and S. Jajodia, “Intrusion detection techniques,” in *The Internet Encyclopedia*, H. Bidgoli, Ed. New York: John Wiley & Sons, 2003.

[7] B. Zhu and S. Sastry, “SCADA-specific intrusion detection/prevention systems: A survey and taxonomy,” in *Proc. of the First Workshop on Secure Control Systems (SCS), CPSWeek 2010*, Stockholm, Sweden, April 2010, pp.381-385.

[8] S. Cheung, B. Dutertre, M. Fong, U. Lindqvist, K. Skinner, and A. Valdes, "Using model-based intrusion detection for SCADA networks," in *Proc. of the SCADA Security Scientific Symposium*, Miami Beach, FL, January 2007, pp.127-134.

[9] D. Yang, A. Usynin, and J. W. Hines, "Anomaly-Based intrusion detection for SCADA systems," in *Proc. of the 5th International Topical Meeting on Nuclear Plant Instrumentation, Control and Human Machine Interface Technologies (MPIC&HMIT 05)*, Albuquerque, NM, November 2006, pp.12-16.

[10] P. Oman and M. Phillips, "Intrusion detection and event monitoring in SCADA networks," in *Critical Infrastructure Protection*,Ed. Boston, MA: Springer, 2007, pp. 161-173.

[11] V. Pothamsetty and M. Franz, Transparent Modbus/TCP Filtering with Linux., July 2010[Online]. Available: http://modbusfw.sourceforge.net/. Accessed 12/30/2011.

[12] R. Chabukswar, B. Sin´opoli, G. Karsai, A. Giani, H. Neema and A. Davis, "Simulation of network attacks on SCADA systems," in *Proc. of the First Secure Control Systems Workshop*, Stockholm, Sweden, April 2010, pp.1-8.

[13] A. Carcano, I. Fovino, M. Masera, and A. Trombetta, "State-based network intrusion detection systems for SCADA protocols: A proof of concept," in *Proc. of the Fourth International Conference on Critical Information Infrastructures Security*, Berlin, Heidelberg, December 2010, pp. 138-150.

[14] A. Giani, G. Karsai, T. Roosta, A. Shah, B. Sinopoli, and J. Wiley, "A testbed for secure and robust SCADA systems," in ACM SIGBED Review, vol. 5, no. 2, pp. 4, 2008.

[15] M. Mallouhi, Y. Al-Nashif, D. Cox, T. Chadaga, and S. Hariri, "A testbed for analyzing security of SCADA control systems (TASSCS)," in *Proc. of the 2011 IEEE PES Innovative Smart Grid Technologies (ISGT)*, Anaheim, CA, January 2011, pp. 1-7.

[16]S. Cohen, W. Sherman, and M. Sherman, "MODBUS/TCP controller for the power supplies in ALS BTS beam line," in *Proc. of the IEEE Particle Accelerator Conference (PAC)*, Albuquerque, NM, June 2007, pp. 425-427.

[17] A. Border and M. Mitzenmacher, "Network applications of bloom filters: A survey," Internet Mathematics, vol. 1, no. 4, pp. 485-509, 2004.

[18] P.S. Almeida, C. Baquero, N. Preguica and D. Hutchison, Scalable Bloom Filters., [Online]. Available: http://gsd.di.uminho.pt/members/cbm/ps/dbloom.pdf Accessed 12/30/2011.

[19] R.B. Bobba, K.M. Rogers, Q. Wang, H. Khurana, K. Nahrstedt, and T.J. Overbye, "Detecting false data injection attacks on DC state estimation," in *Proc. of the First Workshop on Secure Control Systems (SCS)*, CPSWeek 2010, Stockholm Sweden, April 2010, pp. 1-9.

APPENDIX A

MODBUS function codes and descriptions

01- Read Coils
02- Read Discretes
03- Read holding registers
04-Read Input registers
05- Write Coil
06- Write Register
07- Read Exception Status
08- Diagnostics
0B-Get Comm Event Counter
0C-Get Comm Event Log
0F-Write Multiple Coils
10-Write Multiple Registers
11-Report Slave ID
14-Read File Record
15-Write File record
16-Mask Write Register
17-Read/Write Multiple Registers
18-Read FIFO Queue

Hash functions

[1] One-at-a-Time Hash:

```
unsigned int oat( void *key, int len )
 {
  unsigned char *s = key;
  unsigned int hash = 0;
  int i;
  for ( i = 0; i < len; i++ )
  {
   hash += s[i];
   hash += ( hash << 10 );
   hash ^= ( hash >> 6 );
  }

   hash += ( hash << 3 );
   hash ^= ( hash >> 11 );
   hash += ( hash << 15 );
   return hash;
 }
```

[2] Sax Hash:

```
unsigned int sax_hash ( void *key, int len )
{
  unsigned char *s = key;
  unsigned int hash = 0;
  for ( int i = 0; i < len; i++ )
      hash ^= ( hash << 5 ) + ( hash >> 2 ) + s[i];

  return hash;
}
```

[3] FNV hash:

```
Initalize hash to the value of offset_basis

For each octet of data to be hashed

            hash = hash * fnv_prime

            hash = (hash) xor (octet of data)

return hash
```

where hash =‘n’ bit unsigned integer

n =bit length of the hash and

FNV_prime is dependent on ‘n’

VITA

Saranya Parthasarathy received her B.Tech. degree in electrical engineering from the National Institute of Technology (NIT) Tiruchirappalli, India in June 2006. After graduation, she worked as an applications engineer at Oracle India Private Limited, India until July 2009. She joined Texas A&M University in August 2009 to pursue her master's degree in electrical engineering and received her M.S in May 2012.

Saranya Parthasarathy may be reached at:
Department of Electrical and Computer Engineering
322 WERC,
Texas A&M University,
College Station, TX 77843-3128
e-mail: saranya.nitt@gmail.com