Hindawi
Security and Communication Networks
Volume 2021, Article ID 8887666, 15 pages
https://doi.org/10.1155/2021/8887666

Research Article

Detection and Blocking of Replay, False Command, and False Access Injection Commands in SCADA Systems with Modbus Protocol

Rajesh L and Penke Satyanarayana

Department of Electronics and Communication Engineering, Koneru Lakshmaiah Education Foundation, K L Deemed to be University, Vaddeswaram, Guntur 522 502, Andhra Pradesh, India

Correspondence should be addressed to Rajesh L; locharalarajesh@gmail.com

Received 2 September 2020; Revised 21 April 2021; Accepted 9 August 2021; Published 27 September 2021

Academic Editor: Fawad Ahmed

Industrial control systems (ICS) are being used for surveillance and controlling numerous industrial process plants in national critical infrastructures. Supervisory control and data acquisition (SCADA) system is a core component in ICS systems for continuous monitoring and controlling these process plants. Legacy SCADA systems are working in isolated networks and using proprietary communication protocols which made them less exposed to cyber threats. In recent times, these ICS systems have been connected to Internet and corporate networks for data sharing and remote monitoring. They are also using open protocols and operating systems. This leads to vulnerabilities of the system to cyberattacks. Cybersecurity threats are more prevalent than ever in ICS systems. These attacks may be external or internal. Modbus is a widely deployed communication protocol for SCADA communications. There is no security in design of Modbus protocol, and it is vulnerable to numerous cyberattacks. In this paper, we worked for False Command Injection attack, False Access Injection attack, and replay attacks on Modbus protocol. Initially, a real-time SCADA testbed was set up, and we envisaged the impact of these attacks on Modbus protocol data using the testbed. In this work, we used local area network (LAN) environment only for simulating the attacks. We assumed that the attacks penetrated the LAN network. We proposed and developed (a) a method to detect replay attacks by incorporating time stamp and sequence number in Modbus communications and (b) a frame filtering module which will block unauthorized attacks like False Command Injection and False Access Injection attacks to reach programmable logic controller (PLC). Numbers of attacks were simulated and the performance of the method was measured using attack block rate (ABR). It blocked 97% of malicious Modbus transactions or attacks to reach the PLC. It protects SCADA systems from attackers, which is a core component of industrial control systems. The solution enhanced the security of SCADA systems with Modbus protocol.

1. Introduction

Industrial control systems are very crucial for automation of process plants. They are used for surveillance and controlling a plant [1]. These systems are in different forms such as supervisory control and data acquisition (SCADA) systems and distributed control systems (DCS). SCADA system contains data acquisition servers, display-monitoring and controlling clients, historian systems, field instruments, and programmable logic controllers (PLC) [2]. Figure 1 displays a typical SCADA system [3]. It contains different levels of connectivity, systems to differentiate the functionality, and services provided by them. The physical parameters such as pressure, temperature, and pump status will be sensed by suitable sensors and the electrical signal will be passed to PLC through input-output cards. The PLC will collect the data from IO boards, process the data, and send it to human machine interface (HMI) through a suitable communication protocol [4]. In old days, these systems were connected in local area network (LAN). They used mostly proprietary communication protocols. As technology advances, the SCADA data is required for other applications like enterprise resource planning (ERP), energy management system (EMS), and so forth. These systems are connected to Internet

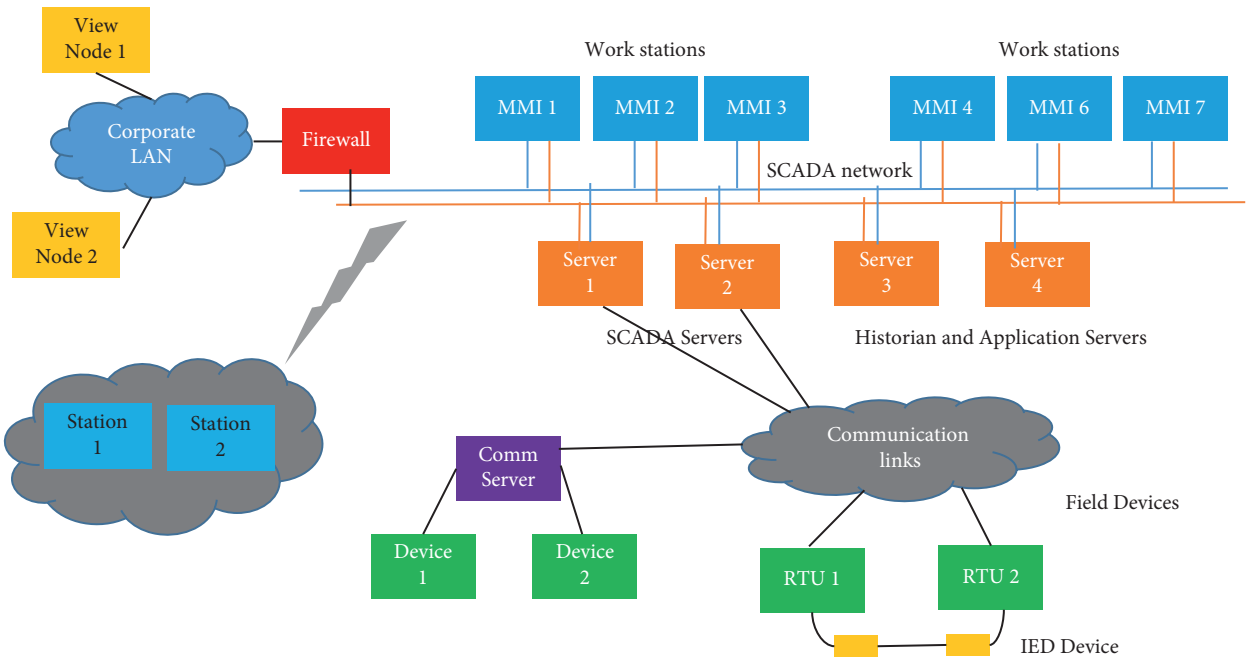

Figure 1: A typical SCADA system architecture [3].

for remote monitoring and data sharing. These systems are also connecting to corporate highly interconnected networks for remote monitoring. The connectivity provides remote surveillance but opens the doors for attackers. Hence, these systems are vulnerable to cyberattacks [5–7]. These attacks may be from internal or external sources.

There were a number of incidents reported all over the world on ICS systems. Some of the examples are Malware Stuxnet attack on SCADA systems in nuclear plants in Iran in 2010, attack on Maroochy Water in 2000, and so forth [8]. The U.S. Department of Homeland Security's (DHS) Industrial Control Systems Cyber Emergency Response Team (ICS-CERT) released a number of attacks reported on ICS system. Every year the attacks are increasing. Even though there are a number of potential areas of vulnerabilities in SCADA components, cyberattacks to communication protocols are a paramount issue [9].

1.1. Modbus Protocol and Vulnerabilities. A SCADA system uses a suitable communication protocol like Modbus, DNP, and so forth for bidirectional data transfer between HMI and PLC [10]. Sensors will sense (measure) the parameters and PLC will collect data from sensors located in the field. This data will be transferred to HMI through a communication protocol [11]. Modbus is most widely deployed communication protocol in control and automation industry. Modbus protocol is an open protocol that was designed in the 1970s [12]. It is a simple request response message protocol at application layer. Modbus is available on two varieties: Modbus Serial and Modbus TCP [13]. The frame format of this protocol is shown in Figure 2. It contains two parts: (i) MBAP Header and (ii) Protocol Data Unit (PDU). The header contains the required details for transmission like

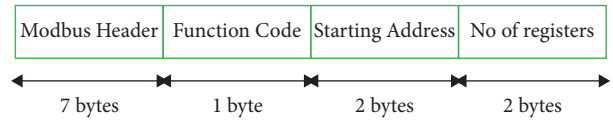

Figure 2: Modbus frame structure [14].

slave ID, protocol ID, and length of the frame to determine the boundaries of the frame. If the physical layer is serial communication, then the nodes participating in the communication are called server and client. They are also called master and slave, if the physical layer is Ethernet (TCP). Modbus protocol contains the following important fields [14]:

(i) Function code

(ii) Starting address of register

(iii) Number of registers

In SCADA communication, data acquisition server or HMI will have Modbus client component (master) and PLC will have its pair component, that is, server component (slave), of Modbus protocol. The master has to send the Modbus request to slave device to poll the data. The slave device will respond to the request with required response and send it to the master. If the request is not correct, then the slave will send exception response to the master. Function code determines the action to be done at Slave. Table 1 gives the details of various function codes and their corresponding actions to be performed. The table contains frequently used function codes in our testing. Starting address and number of registers communicate to the slave how many tags/registers are to be polled by the master [13, 14].

Table 1: Various Modbus function codes and their actions on data.

Function code	Action to be performed
1	Reading of coils
2	Reading of status registers
3	Reading holding registers
4	Reading of input registers
5	Writing to single coil
6	Writing to single holding register
15	Multiple coils: write
16	Multiple holding registers: write

Modbus protocol was designed without considering security aspects because ICS systems were isolated from outside network in earlier days. There was no awareness of cyberattacks when Modbus was designed. Modbus protocol lacks various security features described as follows [15–17]:

(i) Integrity of Modbus frame is not checking by peer devices [18, 19]. The frame can be modified by any attacker and peer devices cannot identify this activity.

(ii) There is no facility for maintaining confidentiality of messages. The Modbus frames will be transferred in plain text and any Man-in-the-Middle attacker can sniff the packet and get the frame information.

(iii) It does not support time stamp for the frames. This is one of the crucial problems because peer devices do not know whether the received response is obtained for the recent query (request) or old query. Any drastic change may happen due to mismatch of real-time field values.

(iv) Modbus is an open protocol and it had simple frame formats. A simple Wireshark tool can be used by attacker to retrieve the information from the network.

As this protocol lacks above security features like integrity, confidentiality, and nonrepudiation, the protocol is more vulnerable to numerous cyberattacks like Man-in-the-Middle attacks in the form of False Command Injection (FCI), False Access Injection (FAI), or False Response Injection; replay attack; and Denial-of-Service (DoS) attacks [20–22]. Out of above attacks, we concentrated on False Command Injection, False Access Injection, and replay attack in this paper. We demonstrated impact of these attacks and proposed and developed methods to block these attacks to reach PLC and HMI. We considered only local area network for simulating the attacks.

1.2. Attacks on Modbus Protocol. Modbus protocol has been suffering from a number of attacks as discussed in Section 1.1. In this research, we worked for False Command Injection, False Access Injection, and replay attacks as follows.

1.2.1. Replay Attacks. The Man-in-the-Middle attacker will store the Modbus messages and will send these messages to target nodes, that is, HMI or PLC, after some time intentionally [23]. As Modbus frame does not have time stamp field, PLC or HMI cannot differentiate whether the response was obtained for recent request frame or old frame. The response in the frame may be old status of field parameters, but HMI will process the frame and erroneous values will be displayed on SCADA mimics. In the same way, the PLC will process the control command and trigger the actuators. This will hamper the operations and may lead to irreversible loss to human life and economy of nation. This attack is very distinctive type that looks like accurate data, but it contains old status [24, 25].

1.2.2. False Command Injection (FCI) Attack. Attacker can send erroneous commands to the PLC like stop the valves while pumping is running, start emergency plant shutdown, and so forth. The attacker can send command to PLC which will disrupt the plant [26]. The aim is execution of arbitrary commands to take control.

1.2.3. False Access Injection (FAI) Attack. The attacker can send false access commands by sending Modbus requests with trial-and-error function codes and starting addresses. It will impose load on the PLC and make it busy. The attacker can send wrong Modbus requests at very high frequency. The PLC will be busy in processing these requests with exception responses. It cannot respond to legitimate HMI commands and requests. This leads to a DoS attack [27].

1.3. Existing Related Works. A number of research scholars worked for security issues of ICS/SCADA systems, communication protocol security, and cyber security of industrial control systems. They also provided some solutions to these problems. The literature survey was conducted in a logical way. Initially, we conducted survey on importance of security of communication protocols in industrial control systems. Next, we described the existing works and shortfalls of those methods to provide security in communication protocols. The next paragraph explains the literature survey on existing works.

Ghosh and Sampalli [28] emphasized the importance of security of SCADA systems. They provided a classification of attacks based on security requirements and network protocol layers. In this paper, they organized security schemes based on current standards, as well as detection and prevention of attacks. They also addressed future challenges in SCADA system security. The authors highlighted that those methods for detection of attacks are more available than methods for prevention of attacks in literature. From this paper, we can understand that it requires to work on prevention of cyberattacks.

Priyanga et al. [29] developed a novel hypergraph-based anomaly detection technique with enhanced principal component analysis and convolution neural network (EPCA-HG-CNN) to detect deviant behaviors of such systems. This method was useful to detect the attacks not for prevention of attacks.

Qian et al. [30] proposed a method based on machine learning techniques where nonparallel hyperplane based fuzzy classifier was used to detect replay and DoS attacks. They informed that hackers even do not need to tamper any data for replay attack; the only thing they need to do is to send the data package eavesdropped from the sensors at another time. The package cannot be detected to be false because there is no time stamp in the protocol used in SCADA systems and the data are in the right form. This method detects only the attacks happening, and it cannot block the attacks. The solution did not provide any remedy for these attacks. Li et al. [31] developed methods that are capable of distinguishing equipment faults from bona fide cyberattacks like replay attack. This method was also used for detection of replay attacks, not for blocking these attacks.

Yusheng et al. [32] explained about various vulnerabilities of Modbus protocol. They proposed real-time deep inspection for Modbus TCP traffic in intrusion detection of industrial control systems based on Modbus protocol. Their method consisted of two modules: rule extraction and deep inspection. The first part separates the Modbus TCP packet into network layer, transport layer, and application layer. The second module generated normal and abnormal rules based on the correlation among those three subparts. The deep inspection module was used for continually correlating the classifications to determine whether the normal or abnormal rule currently applied is a false positive. It is not clear how the rules would be updated over time to detect new attacks or if the rule extraction module could be susceptible to manipulation on the sliding window process due to slow-rate attacks. The paper presented a method to detect the attacks again, not for their prevention.

Fovino et al. [33] developed a secure Modbus protocol based on RSA signature and SHA hash but they did not verify the protocol specific parameters at controller. The frame was transferring in plain text format. They did not address all types of attacks on Modbus protocol. Shahzad et al. [34, 35] developed cryptographic solution using AES, RSA, and SHA for achieving security in Modbus protocol and there was no filtering mechanism. The authors in [36] used AES and hashing for securing IEC 60870-5-104 and Modbus in multicasting polling scenarios. They did not consider integrity of the Modbus frames.

Dudak et al. [37] described enhancement of features of serial Modbus protocol. They developed uBUS protocol by adding some of the protocol specific features. They concentrated on features of Modbus protocol instead of security features and attacks. You and Ge [38] developed Modbus protocol for building security. Erez and Wool [39] developed anomaly detection in Modbus SCADA control registers. The method detects irregular changes in Modbus/TCP SCADA control register values. Hayes and Khatib [40] enhanced security in Modbus by using hash-based message authentication codes and stream transmission control protocol. The authors presented a new method based on stream transmission control protocol (SCTP) and authentication messages based on keyed-hash functions (HMAC) in order to provide a security solution for transactions on Modbus TCP and to establish a mutual authentication mechanism. In this case, all the solutions are software-based and do not include an additional hardware for securing the storage of the secret keys used.

Jung et al. [41] proposed whitelist for SCADA traffic using repetitive communication characteristics of the SCADA system. Kang et al. [42] developed techniques for preparation of whitelist for firewall of SCADA traffic. They provided the concepts of firewall using whitelist generation only. Barbosa et al. [43] proposed approach incorporates a learning phase in which a flow whitelist is learned by capturing network traffic over a period of time and aggregating it into flows. After the learning phase is complete, any nonwhitelisted connection observed generates an alarm. The evaluation of the approach focuses on two important whitelist characteristics: size and stability. The applicability of the approach is demonstrated using real-world traffic traces captured at two water treatment plants and at an electric-gas utility. Serkan et al. [44] studied the FDI attack and simulated the attack in testbed to change the billing information in smart grid application. They explained how this attack can be simulated and proposed methods to detect the attack in the network. They proposed monitoring continuously the ARP and IP changes in the network and alerting the system. This method cannot block or stop to trigger the attack; it will alert the system by critical alarms.

Most of the authors [45–47] worked with testbed using simulation and smart grid domain only. In another section of studies, mathematical modeling [48] and graphical theoretical approach to the network modeling [49] were used for the detection of attacks. In the majority of researches on the security of industrial control systems, no implementation has been done to a real system. Simulation testbed cannot reflect actual real-time system. Zhao and Smidts [50] proposed a two-step hypothesis testing method for detecting and distinguishing replay attacks from other anomalies. The proposed method was demonstrated using a steam generator in a nuclear power plant. This method is suitable for specific domain NPP, not a general model.

From the literature review, it was understood that cyber security of SCADA/ICS systems was a paramount issue in these days. Modbus protocol is vulnerable to cyberattacks; it needs to provide secure data transfer in ICS systems through Modbus protocol. The existing methods were used to detect the attacks not for prevention of these attacks. They did not provide a complete solution to address all the security attacks of Modbus protocol. The FCI, FAI, and replay attacks were less addressed in the literature. The existing methods did not provide end-to-end security in Modbus protocol. In this paper, we analyzed the impact of False Command Injection, False Access Injection, and replay attacks on Modbus protocol using a real-time testbed. After that, we proposed and developed a new method which would provide end-to-end security to detect and block these attacks. We simulated these attacks a number of times and performance was measured using various metrics. In this work, we simulated the attacks in LAN environment. We assumed that the attacks penetrated from outside through Internet connectivity of IT hardware into the LAN (SCADA network). Hence, these attacks were simulated by connecting

the systems in local LAN. Hence, this study is limited to LAN connectivity only. We will work for WAN/Internet in the near future.

The rest of the paper is organized as follows: Section 2 describes the testbed and simulation of attacks on testbed. Section 3 explains the design and development of the method. Section 4 describes the testing and results of the research. The paper is concluded in Section 5.

2. Impact of Cyberattacks on Modbus Protocol

In this section, we described the experimental testbed, simulation of attacks, and impact of these attacks on Modbus communication. We simulated attacks on a real-time testbed for analyzing the impact of these attacks on Modbus.

2.1. Testbed Setup. We set up a real-time testbed in our lab for research. The testbed was shown in Figure 3. It contains one SCADA HMI or DAQ Server with Modbus protocol, one PLC, and one attacker's PC which was used to simulate cyberattacks as *internal* source. These systems are connected in a local LAN switch. We used a Netgear make managed switch for network connectivity. We also connected a web server in the same SCADA LAN for providing web interface of SCADA system. The web server was connected to Internet using a public IP address. A laptop was connected to the Internet and used as *external* attack simulator. The details of the systems are furnished in Table 2. We developed a SCADA application as shown in Figure 4 for testing. The mimic was developed using graphics builder tool of Vijeo Citect SCADA package which was loaded in SCADA HMI. The SCADA package contains Modbus protocol for communicating Schneider make M340 PLC. Initially the PLC was connected directly without gateway PC. The proposed module was running in gateway PC which will protect PLC from the attacks. Detaching the proposed functionality into separate gateway modules will provide an open and standard solution which will be used for existing legacy systems and provides interoperability between multiple manufacturing vendors.

The IP address of PLC can be accessed by (1) accessing host file of SCADA server: IP addresses and domain names are available in host file of SCADA server system. This file lists computer names, IP addresses of SCADA servers, work stations, PLCs, and alias names for routing within the network. (2) Web server was connected to Internet to provide web access and it was connected to SCADA LAN for updating SCADA values. The web server contained public IP address and attacks could access private IP addresses (like PLC, SCADA server) through accessing this public IP address. (3) SCADA systems have been providing remote access through various tools like remote desktop, AnyDesk, and so forth. Attacks can access server, PLC using these tools very easily. Sometimes SCADA systems are distributed geographically over a large space and RTU/PLCs are connecting to SCADA server through GPRS connectivity. Attackers can connect to RTU at site and can destroy the system. Once attacker knows the IP address of PLC by any

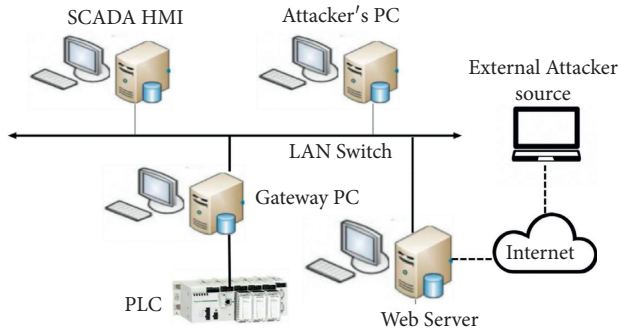

Figure 3: SCADA testbed used for research.

method from the methods mentioned above, the system can be destructed in any manner.

The SCADA application contained two tanks with water. These two tanks were connected using a pump. If the pump starts, then the water will transfer from Tank A to Tank B. We used two Panasonic make level sensors MS-UA11-1 to measure the tank levels. We developed interlock logic in PLC like as follows: Whenever destination tank level reaches 90 cm, the pump will stop automatically by logic. The logic block will continuously monitor destination tank level and it compares it with 90 cm value. If the tank level is more than 90 cm, then it will send pump stop command. The pump will be in state S0 initially. Whenever the destination tank level is more than 90 cm, it will be in state S1. The state diagram was shown in Figure 5. The pump will start automatically once the destination tank reaches zero value. The pump will start whenever the destination tank level reaches 90 cm. But the stop command was sent to pump because of replay attack. The solenoids valves are new ASCO make 24VDC EF8317G35. We used Crompton 0.5HP SP Aquagold 50 Water Pump. These transmitters gave 4 to 20 mA output current and connected to Analog Input and Digital Input modules of PLC. The start and stop commands of motor were controlled by PLC through relay contact which is connected to Digital Output card of PLC.

2.2. Simulation of Attacks. Next, we simulated replay attack, False Command Injection attack, and False Access Injection attack on the testbed from attacker's PC and explained the procedure in subsequent paragraphs.

2.2.1. Replay Attack. The Modbus simulator was loaded in attacker's PC. We used Ettercap tool to scan the target devices and bind the MAC address with target IP address using ARP protocol. Ettercap is a free and open-source network security tool for simulating Man-in-the-Middle attacks in the network. It can be used for computer network protocol analysis and security auditing. Next all messages between PLC and HMI were routed through this attacker's PC. Next, we loaded Wireshark tool to capture the Modbus packet frames. We stored the frame packets and sent them to PLC and HMI after some time.

We observed the impact of this attack on fluid transfer between tanks in the testbed. The graph in Figure 6 shows

Table 2: Details of testbed in lab.

System	Make and model	Specifications	IP address	Other information
SCADA HMI	HP Z420	8 GB RAM with Intel processor	192.168.1.1	Schneider Vijeo Citect SCADA HMI package was loaded in this system
PLC	Schneider make	M340 model	192.168.1.10	PLC logic was written using programming software, Unity Pro
Attacker's PC, web server, and gateway PC	HP Z420	4 GB RAM with Intel processor	192.168.1.3	Modbus simulator was loaded

the normal behavior of pump operations. This graph was a trend plot in SCADA package. Whenever destination tank level reached 90 cm, the HMI sent pump stop command. Figure 7 displays the effect of replay attack. The old values of destination tank level were reported and the pump was stopped intermediately as per interlock logic running in PLC. The destination tank level was reported at HMI as more than 90 cm and the pump start signal was triggered by HMI; but this value was not a real-time current value; it was an old value. But HMI did not differentiate these values and triggered the command. The experiment described the effect of replay attack on Modbus communication.

2.2.2. False Command and Access Injection Attacks. Next, we simulated False Command Injection by sending pump stop command, while fluid was transferring between tanks. Figure 8 shows the graph of the levels of the tanks. When the data values are correct, the level of Tank A will be decreasing and Tank B level will be increasing. The trend of tank level looks like a ramp signal as shown in this figure. Next, we simulated False Command Injection to stop the pump, while fluid was transferring. Then the ramp trend was stopped suddenly as shown in Figure 9 at 11 : 06 : 28. It displayed the impact of FCI attack. In this experiment, the pump was stopped unexpectedly because of simulation of False Command Injection.

From the above testing, it is concluded that Modbus is vulnerable to False Command and Access Injection attacks and replay attacks and it needs to provide a solution for detection of and blocking these attacks. Section 3 explains the proposed and developed solution for this problem.

3. Design of Methods

In this section, we describe our proposed and developed methods for prevention of False Command Injection and False Access Injection attacks and replay attacks. Initially the mathematical modeling of the system was presented. Next, the methodology was proposed and developed.

3.1. Mathematical Modeling

3.1.1. Replay Attacks. In this section, we explain our developed solution to mitigate and detect replay attacks in Modbus protocol. Initially we would like to describe the replay attack in mathematical form. After that, we provide the actual solution.

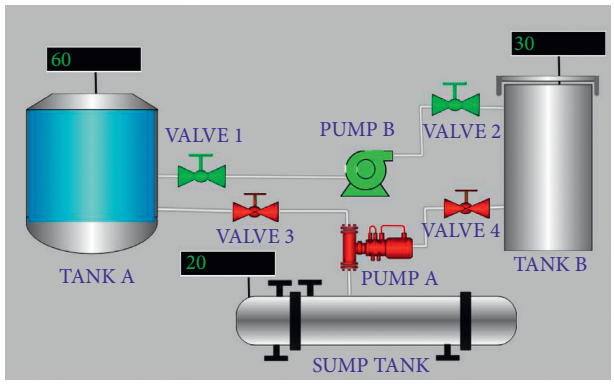

Figure 4: SCADA mimic developed for testing.

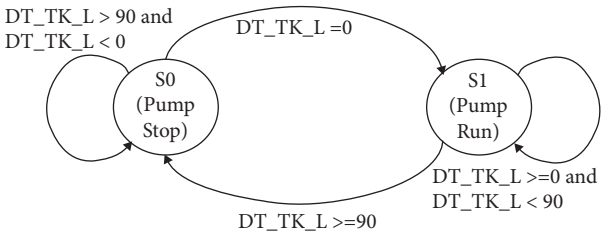

Figure 5: State transition diagram for pump states.

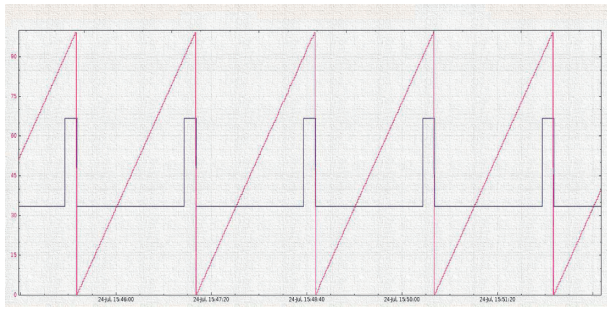

Figure 6: Normal operation of SCADA application.

Let the Modbus request be M_{REQ} containing PLC slave ID, function code, start address, and number of registers to scan.

$$M_{REQ} = \{\text{PLC slave ID, function code, start address, number of reg}\}. \quad (1)$$

This request will be initiated by Modbus client module which will run in SCADA server. SCADA server will poll

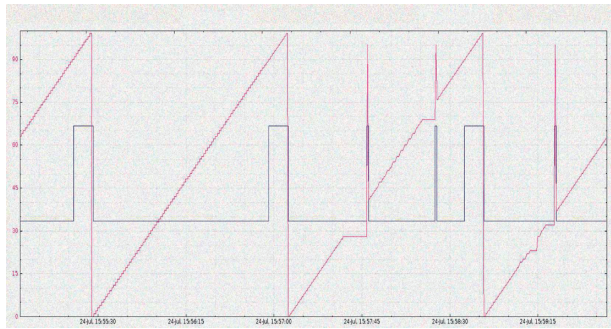

Figure 7: Replay attack on SCADA operation.

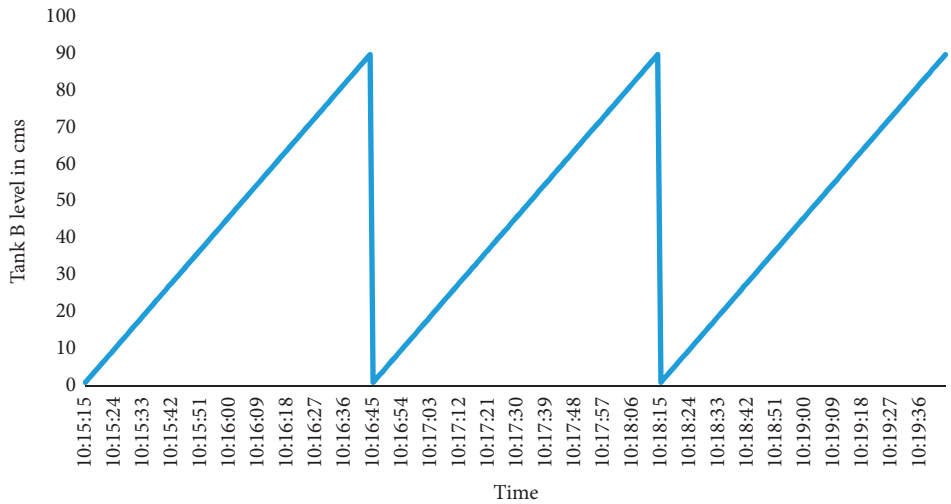

Figure 8: Trend on SCADA of Tank B level during normal fluid transfer.

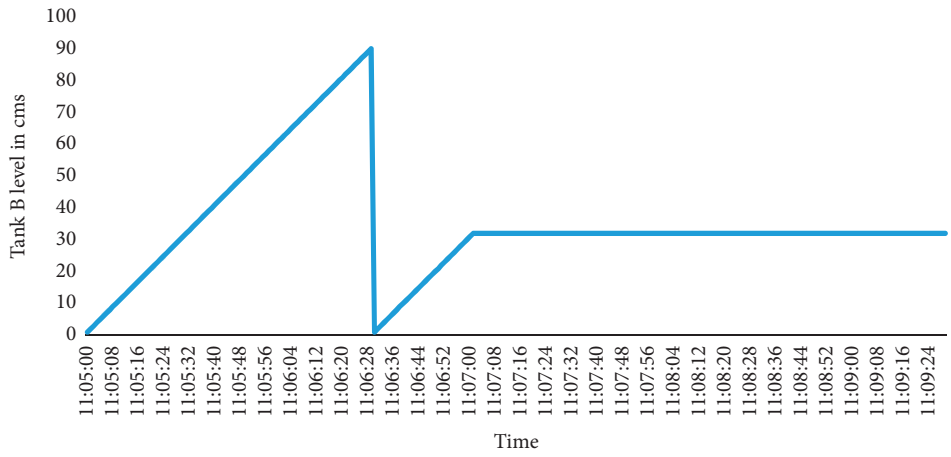

Figure 9: Impact of FCI attack on level of Tank B.

RTU/PLC using Modbus protocol by sending the above-mentioned request to it.

Let the Modbus response to this request be M_{RES} containing the following frame structure as shown in Figure 2:

$$M_{RES} = \{\text{PLC slave ID, function code, frame length, data value}\}. \quad (2)$$

The PLC/RTU will frame the response to the above request and send it to DAQ Server. The DAQ Server will send Modbus requests to PLC like M^1_{REQ}, M^2_{REQ}, and so forth.

$$M_f = \left\{M^i_{\text{Req}},\ i = 0, 1, 2, 3, 4 \ldots\right\}. \quad (3)$$

The PLC will respond to the requests with responses like M^1_{RES}, M^2_{RES}, and so forth.

$$M_r = \left\{M^i_{\text{Res}},\ i = 0, 1, 2, 3, 4 \ldots\right\}. \tag{4}$$

The response will be correct if response for i^{th} query or request is matched with i^{th} response. The response is invalid, if response for i^{th} query is not matched with j^{th} response.

$$M^i_{\text{Req}} \longrightarrow^{\text{yields}} M^i_{\text{Res}}, \quad \text{response is correct,} \tag{5}$$

$$M^i_{\text{Res}} \longrightarrow^{\text{yields}} M^j_{\text{Res}}, \quad \text{response is not correct.} \tag{6}$$

Replay attack detects when (6) occurs at peer device. The sequence number in Modbus request is not the same as sequence number in Modbus response.

To detect the replay attack in Modbus protocol, we included two parameters in the Modbus frame structure:

(a) Sequence number

(b) Time stamp of the frame

We included a sequence number for each and every Modbus frame. The sequence number SEQ_NO_TX of request frame will be copied in response frame by PLC. Modbus client module at HMI will check the sequence number in response frame. If SEQ_NO_RX is matching with one of the SEQ_NO_TX, then it will accept the frame and send it to next step; otherwise, it will reject. Next, it will check the time stamp of response frame. The time difference between request frame and response frame will be calculated and if the difference is more than threshold value it will reject the frame because the response is a delayed frame.

Each and every frame will have a sequence number in the request frame. The PLC will copy this number in response frame. The sequence number is a numerical number (integer) which ranges from 0 to 255 and again it will start from 0. The frame will be changed as follows:

$$\begin{aligned} \text{M}_{\text{REQ}} &= \{\text{Seq no, PLC slave ID, function code, start address, number of reg}\}, \\ \text{M}_{\text{RES}} &= \{\text{Seq no, PLC slave ID, function code, frame length, data value}\}. \end{aligned} \tag{7}$$

The HMI will check the sequence number of response frame. If it matches with the sequence number of request frame, which was sent recently, then it will accept the frame; otherwise, it will reject it.

$$\begin{aligned} &\text{Accept frame if SEQ\_NO\_TX}_{\text{req}} = \text{SEQ\_NO\_RX}_{\text{res}}, \\ &\text{reject frame if SEQ\_NO\_TX}_{\text{req}} \neq \text{SEQ\_NO\_RX}_{\text{res}}. \end{aligned} \tag{8}$$

This method may provide some level of checking but this is not suitable for the following cases:

(a) When multiple requests are sent to PLC without waiting for responses

(b) When the sequence number in response frame matches with request frame but it is after "n" iterations of 255 values

We attended to this problem by implementing time stamp in the frame. Each and every frame will be time-stamped by PLC and HMI. As we discussed above, Modbus did not support time stamp of frame. We add an 8-byte time stamp field to Modbus frame as shown in Figure 10. This time stamp field will be used for checking the freshness of Modbus frame. Generally, Modbus request and responses are periodic in nature. Once the plant was commissioned, the data blocks with tags were fixed. The HMI will poll PLC with fixed configuration for every periodic scan time. HMI will send Modbus requests to PLC with periodic time intervals. If replay attack was launched, then time stamp of the response frame would be very much lag of present time. The module will detect the time gap, reject the frames, and prevent the replay attacks. The threshold for checking the delay of time stamps can be configured by user. The module will automatically adjust this threshold value based on traffic conditions.

In our method, the time delay was calculated between present current time and time stamp of the frame. If the time delay or duration was more than threshold value, the frame was declared as old frame and rejected. The threshold value depends on scanning or polling interval and number of data blocks. If data blocks are high, the PLC will take more time for execution of request. The polling interval may be fixed in some projects, but it may vary and depends on PLC execution time for each query. For example, in our project, HMI polls PLC for every 100 ms for each data block. The data block contains starting address, number of registers, function code, and slave ID. The project contains 4 data blocks for scanning. Then HMI started first data block at 0 ms and PLC took 4 ms for execution of request. HMI received response frame at 64 ms with 60 ms of transmission delay. Next, PLC may wait for 40 ms and initiate next cycle of scanning or may start polling of next data block immediately to achieve fast data acquisition. In the second case, the time duration for one cycle of data blocks is 60 ms in our project. But this time duration is not fixed and it may vary depending on PLC execution time for every request and transmission delay which depends again on data traffic in the network. Hence, it is required to select optimum threshold value for detecting replay attack.

3.1.2. False Command Injection and False Access Injection Attacks. The block diagram for mathematical modeling of the module is shown in Figure 11. Let the input to the frame filtering module be Modbus Frame F_i. It contains Modbus function code (FC), start address (SA), and number of registers (NR) with MBAP Header. It can be expressed as follows:

$$F_i = \{\text{MBAP Header}||FC||SA|NR\}. \tag{9}$$

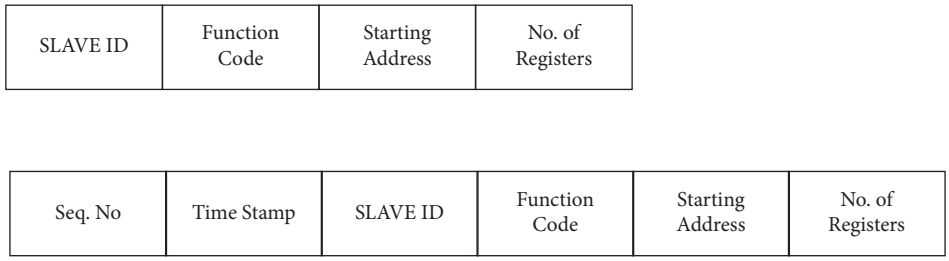

Figure 10: Modbus frame structure, actual and modified.

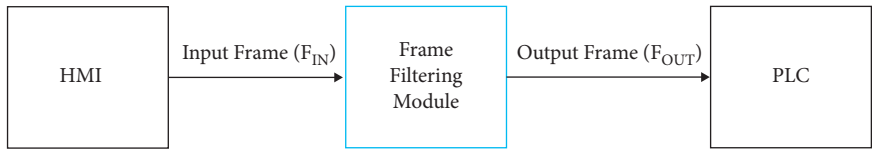

Figure 11: Mathematical modeling of frame filtering module.

The total input to frame filtering module is F_{IN} which is sequence of Modbus frames F_i. The number of frames depends on database in HMI and PLC. Let the number of requests be N. F_{IN} can be expressed as follows:

$$F_{IN} = \{F_1, F_2, F_3, \ldots F_N\}. \tag{10}$$

The module will pass the frame, F_i , if it satisfies the configuration of the module; otherwise, it will block the frame. The output of the frame filtering module is F_o.

$$\begin{aligned} F_o &= F_i \quad \text{if } F_i \text{ matches the configuration,} \\ F_o &= 0 \quad \text{if } F_i \text{ does not match the configuration.} \end{aligned} \tag{11}$$

The total output of frame filtering module is F_{OUT} consisting of F_i or no frame. It can be expressed as follows:

$$F_{OUT} = \{F_1, 0, 0, F_4 \ldots F_N\}. \tag{12}$$

The performance of the frame filtering module can be evaluated by measuring frame rejection rate or attack block rate, which is defined as ratio of number of attacks blocked to number of attacks generated as shown in the following equation:

$$\text{Attack Block Rate (ABR)} = \frac{\text{no.of attacks blocked}}{\text{no.of attacks generated}}. \tag{13}$$

Let the generated attack frames at input of the module be F_g; let the number of blocked attack frames be F_b; then attack block rate can be defined as

$$\%\text{ABR} = \frac{F_b}{F_g} * 100. \tag{14}$$

In this work, we designed a frame filtering module which will send only authorized commands and Modbus requests to PLC. This module will be loaded in a gateway PC. It is basically a computer system. Instead of connecting to network, PLC will be connected to this PC. The communication module of PLC had one RJ45 port for Modbus interface. The port was connected to gateway PC LAN port directly with LAN cable. The destination IP address of Modbus driver at SCADA package was assigned with gateway PC. Hence, instead of connecting to PLC directly, DAQ Server/HMI will connect to this frame filtering module in gateway PC. Generally, every PLC will perform configuration with memory addresses. The memory contains the starting address and number of registers. For example, analog values will use holding registers. The memory range of holding registers contain from 4×00001 to 4×05000. The holding register address shall be between these limits; otherwise, PLC returns exception error.

The flow chart of this module is shown in Figure 12. This module has to be configured using a configuration file. In the configuration of the device, the user has to provide the IP address of the Master device, port number of the connection, allowed function codes, and allowed memory addresses (register addresses). The module will be in listen mode to accept the connection from Master device (connection from DAQ Server/HMI). It will accept the connection if the IP address is an authorized one. It will check the IP address in accept socket call with configured addresses. If the IP address is not matching with any configured IP address, then the connection will be dropped or rejected and the socket will be closed. Next it will check whether the function code is allowed or not. If it is allowed, then it will check whether the starting address and number of registers are within the limits of configured memory or not. If all the parameters are satisfied, then it will pass the request to PLC; otherwise, the request or command will be dropped. In this way, it will protect the PLC from unauthorized or nonconfigured attacks. Modbus has user configurable and future use function codes. Attacker can send malicious commands using these function codes; but the frame filtering module will not accept those commands and block them. Even the Modbus requests with allowed parameters will be accepted and all other requests or commands will be rejected by this module.

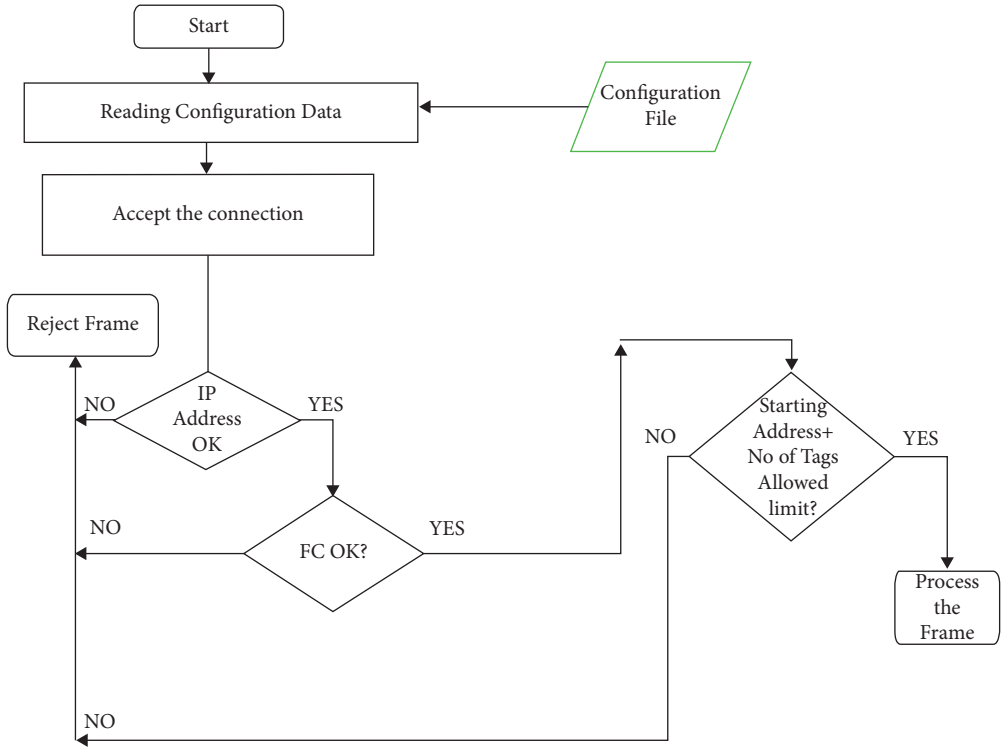

Figure 12: Flow chart of the frame filtering module.

3.2. Open Solution. Even though the above solution provides remedy for the attacks, it changed the frame format of Modbus protocol. Hence, it is not compatible with other manufacturing vendors. It does not provide interoperability. We proposed a solution for this problem. Instead of directly modifying the Modbus frame by DAQ Server or PLC, two gateway modules were placed between DAQ Server and PLC as shown in Figure 13. These gateway modules will modify the frame formats. The gateway at DAQ Server (called gateway1) will add the two new fields to the actual, original Modbus frame (F) initiated by DAQ Server. The modified frame ($F\sim$) will be sent to gateway2 module. This module will remove these fields and send the actual Modbus request (F) to PLC. Then the PLC will frame the response to the request and send it to gateway2 module. Again, gateway2 module will add these new two fields and send them to gateway1. This gateway1 module will remove the fields and send actual response to DAQ Server/HMI. The checking of sequence number and time stamp will be carried out at gateway1 module and gateway2 module instead of checking at DAQ server and HMI. In this way, the gateway modules can be installed in separate PCs and can be easily interfaced with existing legacy systems and any Modbus compliant PLC or device. The gateway2 PC in testbed can be used for this purpose and another PC is required for gateway PC1.

4. Results, Discussion, and Feature Scope

Initially we run the system with modified Modbus without any attack and the system was running normally without any break for 5 hours. Next, we loaded Ettercap tool, Modbus simulator in attacker's PC. We used Ettercap tool for simulating Man-in-the-Middle attack. We sniffed the Modbus packet using Wireshark tool. Using Ettercap tool, we scanned the list of IP addresses in the network. We got IP addresses and MAC addresses of all systems in the network. The attacker's PC used the MAC address from previous scan with ARP spoofing, which resulted in linking of attacker's PC MAC with IP address of HMI/Server. An Ettercap filter can be used to modify Modbus TCP frame originating from HMI and destination to PLC to simulated MITM attack. In our testing, we stored the Modbus requests which are generated by HMI and sent to PLC after various time intervals like 10 ms, 50 ms, 100 ms, 200 ms, 500 ms, 1 sec, 5 sec, 10 sec, 30 sec, 1 min, 30 min, 1 hr, 3 hrs, and 5 hrs as shown in Figure 14. We simulated 100 requests for each time interval. The threshold value was set at 500 ms. The requests received after 500 ms were rejected or dropped by the module. The HMI accepted the Modbus requests with time difference between Modbus frame time stamp and current time being less than 500 ms, and the sequence number matched. From the graph, it was observed that the number of attacks passing through the module is high near to threshold value and the number of attacks was less near to origin because the chance of getting same sequence number at 10 ms is less. The sequence number is a cyclic integer value; hence, it will take $n * t$ ms time to reset it to zero or restart the sequence again. Here "n" is number of Modbus requests and "t" is inter-Modbus request time. The chance of getting same sequence number with less time interval is less; hence, the number of attacks passing through the module is also less.

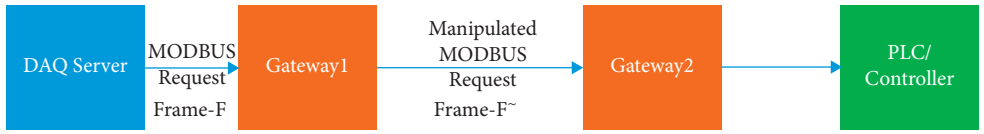

Figure 13: Open solution using gateway modules.

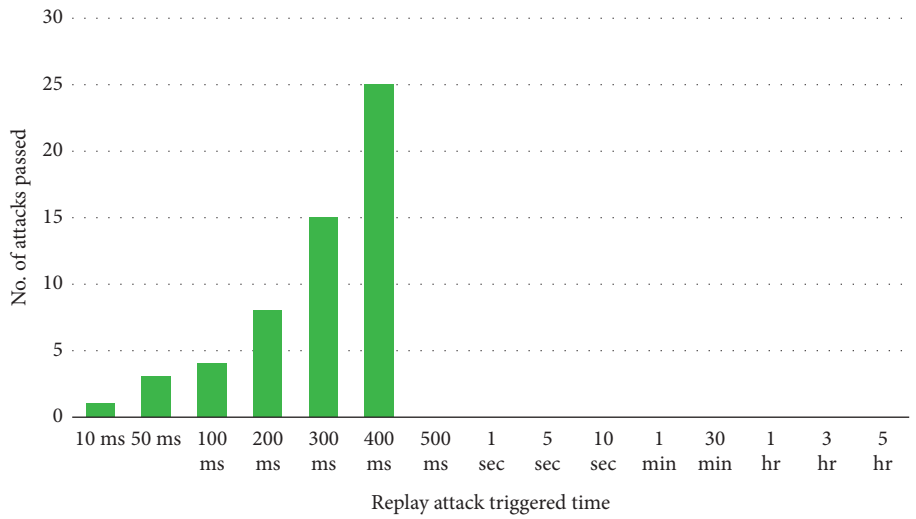

Figure 14: Graph displaying number of replay attacks passing through the module with 500 ms threshold value.

Table 3: Configuration of frame filtering module.

IP address	Function code	Starting address	Number of registers
192.168.1.1	1	0×10000	100
	2	0×10000	250
	4	0×15000	300
192.168.1.2	1	0×10000	200
	4	0×25000	50
192.168.1.3	1	0×02001	4000
	3	0×40000	600
192.168.1.4	2	0×06000	200
	3	0×70000	1250

We configured the frame filtering module with the following configuration as shown in Table 3. It allows Modbus master connections from 4 IP addresses. Each Master will be allowed only some of the function codes and memory addresses. After that, we loaded Modbus simulator in attacker's PC. The simulator can be used for simulating False Command Injection and False Response Injection attacks. We simulated false commands (False Command Injection) using the simulator with function code (FC) = 5, starting address (SA) = 0×0001, and value = 1 (ON). The simulator used IP address of gateway PC as destination IP address. The connection was received at gateway PC. The frame filtering module (FFM) rejected the connection because the IP address (192.168.1.6) was not configured in the module. Next, we simulated the False Command Injection attacks by binding the HMI IP address which is configured in module. We used Ettercap tool to bind the IP address to target device MAC using ARP protocol. The connection was accepted by frame filtering module because IP address of the connection was impersonated by HMI IP address (192.168.1.3). But the frame was rejected as shown in Table 4 because it was not matching the memory configuration of frame filtering module.

Hence, the frame filtering module successfully blocked 97% of attacks. The results can be described as True Negative Rate of 3% and True Positive Rate of 97%. The results are shown in Figure 14 for three iterations. The frame filtering module successfully blocked False Command Injection and False Access Injection attacks to reach the PLC. The HMI and PLC were protected from cyberattacks; hence, the industrial control system was protected from attackers. In this paper, we worked for three attacks on SCADA systems. In the future, we will work for other cyberattacks.

Next, we simulated number of attacks with numerous combinations of function codes and memory addresses. The wrong commands were rejected because of mismatch of configuration. Table 4 displays example of the commands simulated and output of frame filtering module. The table

Table 4: Output of frame filtering module (A: accept, R: reject).

IP address (A/R)	Function code (A/R)	Starting address of registers (A/R)	Number of registers (A/R)	A/R	Reason for rejecting the frame
192.168.1.1 (A)	2 (A)	0×10001 (A)	10 (A)	A	Frame is accepted
192.168.1.2 (A)	5 (A)	0×10005 (A)	1 (A)	A	Frame is accepted
192.168.1.3 (A)	3 (A)	0×40200 (A)	500 (R)	R	Starting address + number of registers crosses the memory access limit
192.168.1.4 (A)	4 (R)			R	Function code 4 is not allowed for this master
192.168.1.6 (R)				R	IP address of master device not matching the list of configured addresses
192.168.1.1 (A)	4 (A)	0×30526(R)		R	The starting address is out of memory access
192.168.1.3 (A)	5 (A)	0×00001	1	R	The memory address is not configured
192.168.1.1 (A)	2(A)	0×10010(A)	25(A)	A	Frame is accepted

displays the reason for rejecting the frame. The Modbus request from IP address 192.168.1.2 was accepted, and the function code and number of registers are within the limit as per configuration; hence, the request was accepted. But the Modbus request from system with IP address 192.168.1.4 was rejected because of configuration mismatch. For some Modbus requests, the IP address and FC were allowed, but the memory access area was out of configured memory; hence, the frame was rejected. Next, we simulated 100, 150, and 200 False Command Injection and False Access Injection attacks in three iterations of each command or request for 10 ms and the frame filtering module blocked 98, 148, and 197 commands, respectively, as shown in Figure 15. It only allowed 2, 2, and 3 commands. The same experiment was conducted 100 times. The mean value of allowed commands in total experiment was 3 commands. We calculated Attack Block Rate (ABR) as given in equation (14). We acquired ABR value of 97%.

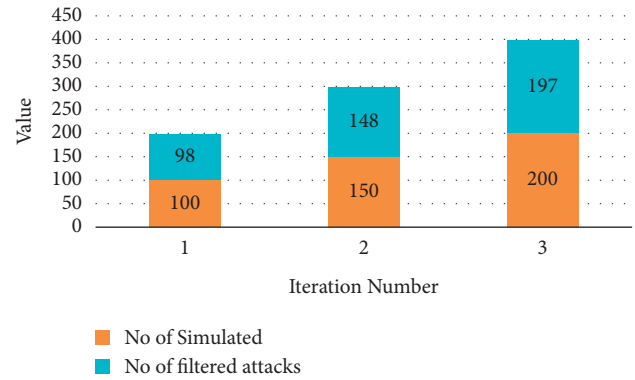

Figure 15: Performance of frame filtering module.

4.1. Overhead of the Solution. This solution provides security in Modbus protocol for secure data transmission between DAQ Server/HMI and PLC, but it adds overhead on Modbus frame length and total time duration. The overhead on Modbus frame length and total time duration are explained in the following sections.

4.1.1. Frame length. We included two fields to the original Modbus frame. The size of sequence number is 2 bytes; time stamp is 8 bytes. Total extra size of frame is 10 bytes. This extra byte size is constant for any Modbus frame. Table 5 indicates how much overhead will be added in the Modbus frame for each type of request.

4.1.2. Time Delay in Total Round Trip Time Duration. As this solution adds processing time for inclusion/deletion of two extra fields and checking of sequence number and time frame, there was a delay in round trip time duration. The total round trip time duration was calculated by time difference between Modbus command triggered at DAQ Server and Modbus response processed at DAQ Server. We conducted the experiment 100 times with different poll intervals from DAQ Server. We got 16 ms average time delay for 1000 ms poll interval. It is 1.6% overhead in total time duration.

Table 5: Overhead of solution on Modbus frame size.

Sl. no.	Type of Modbus request	% overhead on Modbus frame
1	Reading coils/registers	4
2	Writing coils/registers	77

4.2. Comparison of Solution with Existing Related Works. The solution was compared with existing related works. We took references from literature survey and Table 6 describes the differences. Our solution overheads only 1.6% time delay compared to 66% time delay or latency as per [51]. It adds 4% overhead on frame size, compared to 291% as per [33].

In this work, we used testbed with local area network (LAN) connectivity and simulated various attacks by assuming that these attacks penetrated through WAN or Internet into LAN. This study is applicable to internal attacks only. In the future, we will work for external attacks.

Table 6: Comparison of solution with existing related works.

Parameter	Existing works	Our solution
Overhead on frame size/packet length	291% overhead on frame length [[33]]	4% on request, 77% on command frames
Total round trip time duration	66% latency [[51]]	1.6% latency

5. Conclusion

Industrial control systems are using open access networks to leverage efficiency and are more vulnerable to cyberattacks. Modbus is most widely used communication protocol in ICS/SCADA systems, which is one of the core components in Industrial Control Systems. SCADA systems are suffering from security issues and are more vulnerable to cyberattacks. In this paper, we developed an integrated framework solution to prevent the main attacks on SCADA systems. In this framework, we designed a frame filtering module to protect PLC from unauthorized access attacks like False Command Injection and False Access Injection attacks. The module successfully blocked 97% of these attacks. Hence, it protected the PLC from FCI and FAI attacks. We also developed another module in this framework to detect replay attacks in Modbus protocol. The total integrated framework solution was successfully detected and blocked the attacks to reach PLC and HMI. The solution enhances the security for Industrial Control Systems. In this work, we used Local Area Network only for simulating the attacks. We assumed that attacks penetrated through Internet into SCADA network. In the future, we will work for other cyberattacks and WAN/Internet environment.

Abbreviations

SCADA: Supervisory control and data acquisition
ICS: Industrial control systems
PCS: Process control systems
PLC: Programmable logic controllers
RTU: Remote telemetry unit
IED: Intelligent electronic device
LDS: Leak detection system
ERP: Enterprise resource planning
EMS: Energy management system
ICS-CERT: Industrial Control Systems Cyber Emergency Response Team
MAC: Media access control
DAQ: Data acquisition
ARP: Address resolution protocol
PC: Personal computer
FCI: False Command Injection
FAI: False Access Injection.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

The authors would like to acknowledge the management of Koneru Lakshmaiah Education Foundation (K L Deemed to be University) to allow the research in the university.

References

[1] M. Cheminod, L. Durante, and A. Valenzano, "Review of security issues in industrial networks," *IEEE Trans. Ind. Informatics*, vol. 9, pp. 277–293, 2013.

[2] B. Karabacak, S. O. Yildirim, and N. Baykal, "A vulnerability-driven cyber security maturity model for measuring national critical infrastructure protection preparedness," *International Journal of Critical Infrastructure Protection*, vol. 15, pp. 47–59, 2016.

[3] *Pacific Northwest National Laboratory, U.S. Department of Energy, The Role of Authenticated Communications for Electric Power Distribution, Pacific Northwest National Laboratory*, U.S. Department of Energy, Washington, DC, USA, 2006.

[4] K.-C. Kao, W.-H. Chieng, and S.-L. Jeng, "Design and development of an IoT-based web application for an intelligent remote SCADA system," in *Proceedings of the IOP Conference Series: Materials Science and Engineering*, vol. 323, no. 1, pp. 12–25, IOP Publishing Ltd., Komputer, Indonesia, December 2018.

[5] D. Dzung, M. Naedele, T. P. V. Hoff, and M. Crevatin, "Security for Industrial control systems," *Proceedings of the IEEE*, vol. 93, no. 6, pp. 1152–1177, 2015.

[6] B. Babu, T. Ijyas, P. Muneer, and J. Varghese, "Security issues in SCADA based industrial control systems," in *Proceedings of the 2nd International Conference on Anti-cyber Crimes (ICACC)*, pp. 47–51, IEEE, Abha, Saudi Arabia, March 2017.

[7] L. Fillatre, I. Nikiforov, and P. Willett, "Security of SCADA systems against cyber–physical attacks," *IEEE Aerospace and Electronic Systems Magazine*, vol. 32, no. 5, pp. 28–45, 2017.

[8] L. J. Trautman and P. C. Ormerod, "Industrial cyber vulnerabilities: lessons from Stuxnet and the internet of things," *University of Miami Law Review*, vol. 72, p. 761, 2017.

[9] ICS-CERT, "Year in review 2015," 2020, https://ics-cert.us-cert.gov.

[10] W. Su, A. Antoniou, and C. Eagle, "Cyber security of industrial communication protocols," in *Proceedings of the 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1–4, IEEE, Limassol, Cyprus, September 2017.

[11] K. Imtiaz and M. J. Arshad, "Security challenges of industrial communication protocols: threats vulnerabilities and solutions," *International Journal of Computer Science and Telecommunications*, vol. 10, no. 4, 2019.

[12] *Modbus over Serial Line Specification and Implementation Guide V1.02*, 2006, http://www.Modbus-IDA.org.

[13] *Modbus Messaging on TCP/IP Implementation Guide V1.0b*, 2006, http://www.Modbus-IDA.org.

[14] *Modbus Application Protocol Specification V1.1b3*, 2012, http://www.Modbus-IDA.org.

[15] R. Nardone, R. J. Rodríguez, and S. Marrone, "Formal security assessment of Modbus protocol," in *Proceedings of the 2016 11th International Conference for Internet Technology and Secured Transactions (ICITST)*, pp. 142–147, IEEE, Barcelona, Spain, December 2016.

[16] A. Volkova, M. Niedermeier, R. Basmadjian, and H. D. Meer, "Security challenges in control network protocols: a survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 619–639, 2019.

[17] L. Rosa, M. Freitas, S. Mazo, E. Monteiro, T. Cruz, and P. Simões, "A comprehensive security analysis of a SCADA protocol: from OSINT to mitigation," *IEEE Access*, vol. 7, Article ID 42156, 2019.

[18] A. M. Abdul and S. Umar, "Data integrity and security [DIS] based protocol for cognitive radio ad hoc networks," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 5, no. 1, pp. 187–195, 2017.

[19] K. Rambabu and N. Venkatram, "Contemporary affirmation of security and intrusion handling strategies of internet of things in recent literature," *Journal of Theoretical and Applied Information Technology*, vol. 96, no. 9, pp. 2729–2744, 2018.

[20] S. Bhatia, N. Kush, C. Djamaludin, A. Akande, and E. Foo, "Practical modbus flooding attack and detection," in *Proceedings of the Twelfth Australasian Information Security Conference (AISC 2014)[Conferences in Research and Practice in Information Technology, Volume 149]*, pp. 57–65, Australian Computer Society, Inc., Auckland, New Zealand, January 2014.

[21] A. M. Abdul and S. Umar, "Attacks of denial-of-service on networks layer of OSI model and maintaining of security," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 5, no. 1, pp. 181–186, 2017.

[22] L. Rajesh and P. Satyanarayana, "Detecting flooding attacks in communication protocol of industrial control systems," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 1, 2020.

[23] Y. Hu, A Yang, L. Hong, Y. Sun, and L. Sun, "A survey of intrusion detection on industrial control systems," *International Journal of Distributed Sensor Networks*, vol. 14, no. 8, Article ID 1550147718794615, 2018.

[24] Y. Yang, K. McLaughlin, T. Littler et al., "Man-in-the-middle attack test-bed investigating cyber-security vulnerabilities in smart grid SCADA systems," in *Proceedings of the International Conference on Sustainable Power Generation and Supply (SUPERGEN 2012)*, p. 138, Hangzhou, China, September 2012.

[25] V. Somu, D. B. K. Kamesh, J. K. R. Sastry, and S. N. M. Sitara, "Snort rule detection for countering in network attacks," *Advances in Intelligent Systems and Computing*, vol. 515, pp. 573–583, 2017.

[26] B. Chen, N. Pattanaik, A. Goulart, L. Karen, B. Purry, and D. Kundur, "Implementing Attacks for Modbus/TCP Protocol in a Real-Time Cyber Physical System Test bed," in *Proceedings of the 2015 IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR)*, pp. 1–6, IEEE, Charleston, SC, USA, May 2015.

[27] P. Maynard, K. McLaughlin, and B. Haberler, "Towards understanding man-in-the-middle attacks on IEC 60870-5-104 SCADA networks," in *Proceedings of the 2nd International Symposium for ICS & SCADA Cyber Security Research 2014 (ICS-CSR 2014)*, pp. 30–42, St. Poelten, Austria., April 2014.

[28] S. Ghosh and S. Sampalli, "A survey of security in SCADA networks: current issues and future challenges," *IEEE Access*, vol. 7, Article ID 135812, 2019.

[29] P. Priyanga, K. Krithivasan, S. Pravinraj, and V. S. Sriram, "Detection of cyberattacks in industrial control systems using enhanced principal component analysis and hypergraph-based convolution neural network (EPCA-HG-CNN)," *IEEE Transactions on Industry Applications*, vol. 56, no. 4, pp. 4394–4404, 2020.

[30] J. Qian, X. Du, B. Chen, B. Qu, K. Zeng, and J. Liu, "Cyber-physical integrated intrusion detection scheme in SCADA system of process manufacturing industry," *IEEE Access*, vol. 8, Article ID 147471, 2020.

[31] D. Li, N. Gebraeel, and K. Paynabar, "Detection and differentiation of replay attack and equipment faults in SCADA systems," *IEEE Transactions on Automation Science and Engineering*, 2020.

[32] W. Yusheng, K. Fan, Y. Lai et al., "Intrusion detection of industrial control system based on Modbus TCP protocol," in *Proceedings of the IEEE 13th International Symposium on Autonomous Decentralized System (ISADS)*, pp. 156–162, IEEE, Bangkok, Thailand, March 2017.

[33] I. N. Fovino, A. Carcano, M. Masera, and A. Trombetta, "Design and implementation of a secure modbus protocol," *IFIP Advances in Information and Communication Technology Critical Infrastructure Protection*, vol. III, pp. 83–96, 2009.

[34] A. Shahzad, M. Lee, Y.K. Lee et al., "Real time MODBUS transmissions and cryptography security designs and enhancements of protocol sensitive information," *Symmetry*, vol. 7, no. 3, pp. 1176–1210, 2015.

[35] A. Shahzad, S. Musa, and M. Irfan, "Security solution for SCADA protocols communication during multicasting and polling scenario," *Trends in Applied Sciences Research*, vol. 9, no. 7, pp. 396–405, 2014.

[36] X. Luo and Y. Li, "Security enhancement mechanism of modbus TCP protocol," in *Proceedings of the DEStech Transactions on Computer Science and Engineering ICITI*, Seattle, WA, USA, 2018.

[37] J. Dudak, G. Gaspar, S. Sedivy, P. Fabo, L. Pepucha, and P. Tanuska, "Serial communication protocol with enhanced properties–securing communication layer for smart sensors applications," *IEEE Sensors Journal*, vol. 19, no. 1, pp. 378–390, 2019.

[38] W. You and H. Ge, "Design and implementation of modbus protocol for intelligent building security," in *Proceedings of the IEEE 19th International Conference on Communication Technology (ICCT)*, pp. 420–423, IEEE, Xi'an, China, October 2019.

[39] N. Erez and A. Wool, "Control variable classification, modeling and anomaly detection in Modbus/TCP SCADA systems," *International Journal of Critical Infrastructure Protection*, vol. 10, pp. 59–70, 2015.

[40] G. Hayes and K. E. Khatib, "Securing modbus transactions using hash-based message authentication codes and stream transmission control protocol," in *Proceedings of the 2013 Third International Conference on Communications and Information Technology (ICCIT)*, pp. 179–184, IEEE, Beirut, Lebanon, June 2013.

[41] W. Jung, J. Yun, S. Kim, K. Shim, and M. Kim, "Structured whitelist generation in scada network using prefixspan algorithm," in *Proceedings of the 19th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, p. 326, September 2017.

[42] D. H. Kang, B. K. Kim, N. C. Jung, and K. S. Jhang, "Whitelist generation technique for industrial firewall in SCADA networks," in *Frontier and Innovation in Future Computing and Communications*, pp. 525–534, Springer, Dordrecht, Netherland, 2014.

[43] R. R. R. Barbosa, R. Sadre, and A. Pras, "Flow whitelisting in SCADA networks," *International journal of critical infrastructure protection*, vol. 6, no. 3-4, pp. 150–158, 2013.

[44] H. H. G. Serkan, S. Useyin, Y. N. Ercan, U. Furkan, and K. Gokce, "False Data Injection Attacks and the Insider Threat in ¨ Smart Systems," *Computers & Security*, vol. 97, 2020.

[45] D. Myers, S. Suriadi, K. Radke, and E. Foo, "Anomaly detection for industrial control systems using process mining," *Computers & Security*, vol. 78, pp. 103–125, 2018.

[46] Y. Li and Y. Wang, "False data injection attacks with incomplete network topology information in smart grid," *IEEE Access*, vol. 7, pp. 3656–3664, 2019.

[47] X. Liu, P. Zhu, Y. Zhang, and K. Chen, "A collaborative intrusion detection mechanism against false data injection attack in advanced metering infrastructure," *IEEE Transactions on Smart Grid*, vol. 6, no. 5, pp. 2435–2443, 2015.

[48] O. Kosut, L. Jia, R. J. Thomas, and L. Tong, "Malicious data attacks on the smart grid," *IEEE Transactions on Smart Grid*, vol. 2, no. 4, pp. 645–658, 2011.

[49] A. Anwar, A. N. Mahmood, and Z. Tari, "Identification of vulnerable node clusters against false data injection attack in an AMI based smart grid," *Information Systems*, vol. 53, pp. 201–212, 2015.

[50] Y. Zhao and C. Smidts, "A control-theoretic approach to detecting and distinguishing replay attacks from other anomalies in nuclear power plants," *Progress in Nuclear Energy*, vol. 123, Article ID 103315, 2020.

[51] M. K. Ferst, H. F. M. D. Figueiredo, G. Denardin, and J. Lopes, "Implementation of secure communication with modbus and transport layer security protocols," in *Proceedings of the 2018 13th IEEE International Conference on Industry Applications (INDUSCON)*, pp. 155–162, IEEE, São Paulo, Brazil, November 2018.