Symmetry **2015**, *7*, 1176-1210; doi:10.3390/sym7031176

OPEN ACCESS

symmetry

ISSN 2073-8994

www.mdpi.com/journal/symmetry

Article

Real Time MODBUS Transmissions and Cryptography Security Designs and Enhancements of Protocol Sensitive Information

Aamir Shahzad [1], Malrey Lee [1,*], Young-Keun Lee [2,*], Suntae Kim [3], Naixue Xiong [4], Jae-Young Choi [5] and Younghwa Cho [5]

1 Center for Advanced Image and Information Technology,School of Electronics & Information Engineering, Chonbuk National University, 664-14, 1Ga, Deokjin-Dong, Jeonju, Chonbuk 561-756, Korea; E-Mails: mail2aamirshahzad@gmail.com or malikaamirawan2@hotmail.com

2 Department of Orthopedic Surgery, Chonbuk National University Hospital, Jeonju, Jeonbuk 561-721, Korea

3 Department of Software Engineering, Chonbuk National University, 664-14, 1Ga, Deokjin-Dong, Jeonju, Chonbuk 561-756, Korea; E-Mail: stkim@jbnu.ac.kr

4 School of Computer Science, Colorado Technical University, 4435 North Chestnut Street, Colorado Spring, CO 80907, USA; E-Mail: xiongnaixue@gmail.com

5 College of Information and Communication Engineering, Sungkyunkwan University, Suwon 440-746, Korea; E-Mails: jaeychoi@skku.edu (J.-Y.C.); choyh2285@skku.edu (Y.C.)

***** Authors to whom correspondence should be addressed; E-Mails: mrlee@jbnu.ac.kr (M.L.); trueyklee@naver.com (Y.-K.L.); Tel.: +82-10-3611-8004 (M.L.); +82-10-4650-2935 (Y.-K.L.).

Academic Editors: Young-Sik Jeong, Laurence T. Yang and Stefanos Gritzalis

Received: 28 February 2015 / Accepted: 18 June 2015 / Published: 2 July 2015

Abstract: Information technology (IT) security has become a major concern due to the growing demand for information and massive development of client/server applications for various types of applications running on modern IT infrastructure. How has security been taken into account and which paradigms are necessary to minimize security issues while increasing efficiency, reducing the influence on transmissions, ensuring protocol independency and achieving substantial performance? We have found cryptography to be an absolute security mechanism for client/server architectures, and in this study, a new security design was developed with the MODBUS protocol, which is considered to offer phenomenal performance for future development and enhancement of real IT infrastructure. This study is also considered to be a complete development because security is tested in almost all ways of MODBUS communication. The computed measurements are evaluated to validate the overall development, and the results indicate a substantial improvement in security that is differentiated from conventional methods.

Keywords: symmetric encryption; asymmetric encryption; hashing; cryptography buffer

1. Introduction

The MODBUS protocol is part of the supervisory control and data acquisition (SCADA) system, and it is the most commonly used protocol in industrial systems, including the oil and gas industries and power industries [1–4]. The MODBUS protocol offers an application layer (open system interconnection (OSI) model) messaging protocol that constructs the message with implicit function codes and defines communication rules for control systems to supervise and control the overall industrial infrastructure [1,5–7]. Massive progress has been made in-terms of improvements in collecting and analyzing and developing system controls. As a result, MODBUS has become more prominent in industrial applications and is now employed all over the world [8–12].

The MODBUS protocol typically has two types of communication principals, including MODBUS serial line and MODBUS Transmission Control Protocol/ Internet Protocol (TCP/IP). In the MODBUS serial protocol, protocol messages are transmitted between the main controller and the sub-controller and/or vice versa by employing remote terminal unit (RTU) controller modes over the serial lines. Usually, the MODBUS message contains three main fields, including recipient address, protocol data unit (PDU) and error checking field. During transmission, the sub-controller address is added in the specified field in the request message and the corresponding address is placed in a response message that identifies the main controller [1]. Nowadays, the MODBUS protocol provides facilities to establish a connection with a local area network (LAN), and the main controller may be connected to a number of sub-controllers for communication to take place via transport control protocol (TCP). In addition, MODBUS protocol also employs IP interconnectivity between multiple main controllers and sub-controllers, which means that one sub-controller or field device concurrently responds to multiple main controllers and/or multiple sub-controllers are configured with a single main controller in the MODBUS TCP/IP network. During communication, the MODBUS PDU is encapsulated in the TCP payload, and therefore, the MODBUS application protocol (MBAP) is added with the original MODBUS application PDU, which is used in the MODBUS serial protocol [1,4–8].

The MODBUS TCP/IP protocol provides a considerable efficiency for industrial SCADA systems and infrastructure, and the number of field devices that are connected with one or more main controllers via TCP/IP protocols may be geographically located at a distance [1,5,13–15]. Nevertheless, the increased connectivity over different IP based non-proprietary networks and the implementation of an open TCP protocol over an Internet connection results in the MODBUS protocol becoming vulnerable to several types of security attacks [1,5,8,12,15]. In general, little attention has been paid to security for the MODBUS protocol, that is, the MODBUS protocol was designed with maximum functionalities but with little attention to security issues [14–21].

References [8,12,14,22–26] provide details of a survey conducted for SCADA protocol security issues, in addition to potential attacks that are considered to be harmful for SCADA/MODBUS communication [22]. In general, potential attacks that are considered to be harmful for a SCADA/MODBUS system (or network) are grouped into three types: attacks against the MODBUS

protocol specifications, attacks against MODBUS protocol vendor implementations, and attacks against infrastructure or SCADA/MODBUS system components [22,26]. With respect to the initial stage, attacks can be categorized into four main parts based on MODBUS protocol communication, including interception, interruption, modification and fabrication [22,24]. In other words, the MODBUS serial protocol communication includes components such as the main controller, sub-controllers, serial link and messages that may suffer from attacks. In the case where the MODBUS TCP is used, attacks may affect the main controller, sub-controllers, network communication paths and messages. As a result, a detailed taxonomy of the attacks has been conducted, and the attacks to the integrity, authentication, confidentiality and others [22,25,27] are considered to be the most harmful for SCADA/MODBUS communications [23–32].

As a result, SCADA/MODBUS communications have suffered from potential vulnerabilities and attacks that have disrupted service by the protocol as well as the overall SCADA industrial systems [21,22,33–40]. Nowadays, security is considered to be a big challenge for the MODBUS protocol as part of SCADA communication. Real attention is needed to resolve these potential security issues of SCADA/MODBUS communication, and an intelligent mechanism is required for security performance to significantly improve. The core security mechanisms that have been considered to provide security for traditional networks as well as for critical networks, including SCADA/MODBUS network, involve the use of cryptography, such as symmetric and asymmetric encryption algorithms [35–40].

Symmetric key mechanisms are rigorous approaches that improve the security of SCADA/MODBUS messages, and these methods are provide reliable security, even in the case where SCADA needs to carry larger amounts of data in a short session. Usually, an integrated key or session key is employed to secure SCADA communication against attacks, and distribution methods including centralized and decentralized key distribution are used to distribute keys securely using channels between the main controller and sub-controllers (or field devices) and/or vice versa. Several forms of distribution and management for keys have been developed [41–46]. Decentralized key distribution is considered to be the most reliable scheme in which the master keeps control of the Key distribution center (KDC), and SCADA transmission occurs between the main controller and the sub-controller(s) [16,41–47]. However, this study employed a static method or keys were generated and distributed statistically between participating nodes, such as the main controller and the sub-controllers. The certificate authority (CA) is a limitation of this study and should be considered in order to deploy and employ as a future prospect.

Cryptography has been considered to provide the most accurate method to improve security in traditional and in real-time networks [25,29–31,35–40]. These security schemes have several advantages relative to other well known security approaches, including Secure Sockets Layer/Transport Layer Security (SSL/TLS), Internet Protocol Security (IPSec), Secure Shell (SHH), security patterns and others. These cryptography schemes also offer complete security solutions without any other protocol dependencies [25,29–32]. However, the number of key points should be taken into account during security development for real-time networks, such as for SCADA systems [29,30]. Asymmetric cryptography uses a number of keys and extensive computation time relative to symmetric key encryption. Therefore, this could be considered to be an inadequate security solution for a few scenarios of SCADA systems [25,27–32]. Existing end-to-end studies [15–17,19–22,25,28–33] have been conducted on cryptography to improve the security of a SCADA system and its protocols security. In the end-to-end scenarios, the first messages are generated and communication rules are specified from the

SCADA protocols, such as MODBUS, Distributed network protocol 3 (DNP3) and Fieldbus, and these are transmitted to target devices using TCP/IP protocols over the Internet. SCADA protocols are usually specified as proprietary protocols but open connectivity with TCP/IP protocols and/or with other networks has made these (protocols) non-proprietary.

Open connectivity is important for MODBUS protocol to fulfill the demands for communication of end users and SCADA-based industrial equipment. However, TCP/IP protocols also suffer from several vulnerabilities to attacks, such as denial of service (DoS), packet sniffing, spoofing, process table, TCP sequence number generation, IP half scan attacks and others [22–32,48,49]. Security mechanisms, including SSL/TLS, SHH and IPSec, have been employed but these solutions have a limitation in terms of the communication protocol dependencies and security dependencies of the cryptography that has been implemented [25,29,30].

We can conclude from the above security analysis that an inclusive security mechanism is required to not only address the end-to-end phenomenon but also to provide security improvements and to also improve the outlook on security in the future. In this study, a cryptography mechanism is used as an inclusive security solution(s) and is deployed in the MODBUS message protocol before the transmission is sent over open protocols (such as TCP/IP) or other networks. We scrutinized the proposed security measure, and found it to be adequate in replacing other end-to-end solutions. Symmetric and asymmetric algorithms were selected from the most prominent algorithms in the area of cryptography. Although end users could be able to deploy and test other cryptography algorithms, we employed Advanced Encryption Standard (AES), Rivest-Shamir-Adleman cryptosystem (RSA) and Secure Hash Algorithm (SHA-2) algorithms in this study. The main goal of this study is to deploy and test security for every possible communication over the SCADA/MODBUS protocol. This study therefore has the following main goals.

i. To develop an inclusive security solution for the MODBUS messaging protocol, the original MODBUS protocol design is used, and the security functions were deployed in the protocol messaging stack before transmission over open networks. A new cryptography buffer (CB) was designed, deployed and configured for use with the MODBUS protocol messages during open connectivity and during transmission, meaning that CB is employed on both sides of the communication and its function fields are integrated with messages during transmission.
ii. The CB contains a number of fields that are used to keep track of the security developments as well as the MODBUS messaging details. Several intelligent functions have been employed to monitor security developments and sensitive information during transmission.
iii. Security is an important part of MODBUS protocol communications, such as unicasting, broadcasting and multicasting (treated as optional). Therefore, security has been designed according to the communication requirements of the SCADA/MODBUS protocol. With respect to security development, cryptographic algorithms are designed according to the given requirements without affecting communication(s), and the corresponding changes are also made in the cryptography buffer (CB) in order to achieve the desired goals.
iv. MODBUS attack scenarios are created in order to test the level of security, and built-in predominant attack tools were employed for a potential attack to detect attacks in transmissions and to examine the corresponding security.

v. The results of the security performance were computed, analyzed and compared against existing end-to-end security developments, and these were also compared in the absence of the security developments.

The rest of this research paper is organized as follows. Section 2 describes the MODBUS protocol, and Section 3 describes MODBUS messaging on TCP/IP. In Section 4, a security development is made with formal proof, and the cryptography buffer is deployed in Section 5. The testing setup is established and the performance is computed in Section 6. Section 7 provides a comparison, and the significance of the study is discussed in Section 8. Section 9 provides the conclusion and future research.

2. SCADA MODBUS Protocol

MODBUS is one of the prominent protocols used in SCADA communication, and it provides services for message exchange between field devices and other SCADA system applications, such as human-machine interface (HMI) and/or user made software. In general, MODBUS protocol is situated in layer-7 of the OSI model, and it provides an application layer for message services that has also been designated as an application layer messaging protocol. Typically, MODBUS provides request and response message services for the SCADA client/server architecture and whatever media access control (MAC) has been employed at the data link layer (or layer-2) of the seven-layer OSI model [1,5–7]. In the SCADA/MODBUS client/server architecture, four types of message services are used between field devices as follows [1,6]:

i. MODBUS Request Messages, usually the main controller initiates the transmission and sends the request message to the sub-controllers or field devices in the SCADA system.
ii. MODBUS Response Messages, field devices are configured to generate and transmit a response back to the main controller that has requested local circumstances.
iii. MODBUS Message Confirmation, upon receiving a response at the site of the main controller, a confirmation message is transmitted to the sub-controller(s).
iv. MODBUS Message indications, field controllers generate indication messages that show that the request messages have been received.

In the above message services, we used the client as a main controller (MC) and servers as sub-controllers (SCs). In the SCADA system, a request is usually generated at the main controller site (or server site) and a response should be transmitted from the site of the sub-controller(s). The sub-controllers are directly/in-directly connected with the physical world using sensors, actuators and other programmable logical controllers (PLCs) [1,5–7]. The MODBUS client/server communication is visualized in Figure 1.

For layer 7, the MODBUS protocol requires some additional assistance in its lower layers in sequence to transmit the message. In general, the MODBUS messaging protocol employs a master/slave two-layer protocol that transmits data bytes in a serial format in a half duplex mode over modem links, such as RS-232, RS-485 (or Bell 202 or MAP). Nowadays, the MODBUS protocol employs TCP/IP protocols and Ethernet technology to transmit data between the main controller and the sub-controller(s) and vice versa. TCP/IP provides an interaction of the facilities and makes it possible for the MODBUS frames to travel over the routed networks. Here issues may arise as a result of the connectivity between the MODBUS protocol and the TCP protocol. In order to resolve this issue, an additional layer has been added

to map the MODBUS application layer to the TCP protocol. This additional layer (or sub-layer, as used in the original documentation of the MODBUS protocol) performs a function that encapsulates the MODBUS protocol data unit (PDU) into a TCP/IP frame. Thus, the MODBUS PDU travels as a TCP/IP packet over the transmission media and/or the Internet [1,5–7]. The connectivity of the MODBUS protocol with the TCP/IP protocol is shown in Figure 2.

For example, the preliminary main controller initiates communication with the connected sub-controller (SC). Then, the MODBUS messaging protocol generates a message or a protocol data unit (PDU) that contains function code and data requests. For Layer 4, the PDU is encapsulated into the TCP/IP packet that makes it possible for the MODBUS PDU to travels over the network (in a client/server architecture). At the data link layer, the PDU is converted into an application data unit (ADU) by adding some network fields, such as the corresponding network addresses. At the other side, a request message is received and a response message is then generated by employing the same phenomena for the main controller (MC). The error detection codes are added in the request/response APU at Layer 2, which performs error detection during transmission [1,6].

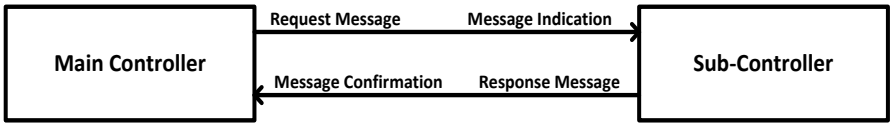

Figure 1. MODBUS Client/Server Communication.

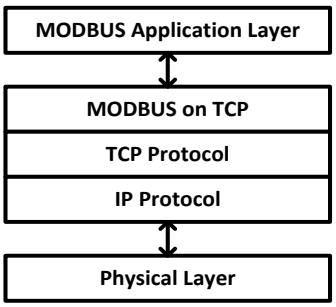

Figure 2. MODBUS Protocol Stack with an Internet Protocol Suite.

MODBUS Protocol Message Structure

For the MODBUS messaging protocol, a protocol data unit (PDU) is generated by adding function code and requesting data. Then, the data is further converted to an application data unit (ADU) by adding two more fields: an address field and an error check field [1,6]. Figure 3 shows the detailed fields of the MODBUS protocol with occupied bytes.

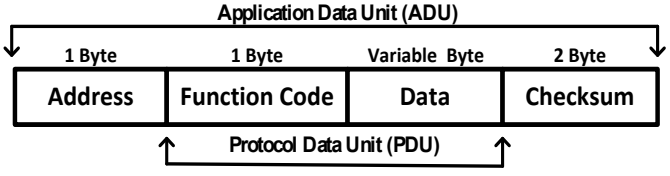

Figure 3. MODBUS Protocol PDU and ADU.

i. **Address Field:** This field contains one byte of information and is designated as the first field of the request/response frames in the MODBUS protocol. The addresses, including that for the main controller and sub-controllers, are specified to identify the controllers for which the request/response is being directed to or from. The address range from 1 to 247 is allocated for each controller. However, the addresses are limited according to the network demands (or by the number of nodes configured in network). Usually, one main controller and 2–3 sub-controllers have been configured at a time for the MODBUS protocol implementation [1,6]. MODBUS defines four basic data types, including coils, discrete inputs, input registers and holding registers, and the address range for these data types is listed in Table 1.

ii. **Function Field:** This field is considered as an important field in the MODBUS protocol frame, and the purpose of the message or frame is defined in this field through the use of various functions, such as read input status, read output status, and others. The required function code is added in the request message (or frame) that identifies the meaning of the message, and an operation can be performed at the target device. At the sub-controller, if the connected PLC or sensor can perform the operation enclosed in the main controller request message, then the response frame echoes its function code according to the request message. If this is not possible, a request message function field will be echoed, plus one is set as the most significant bit [1,6].

Table 1. MODBUS Protocol Data Types and Corresponding Description.

Data Type	Absolute Addresses	Relative Addresses	Description
Coils	00001–09999	0–9998	Read coil status
Coils	00001–09999	0–9998	Force single coil
Coils	00001–09999	0–9998	Force multiple coils
Discrete inputs	10001–19999	0–9998	Read input status
Input registers	30001–39999	0–9998	Read input registers
Holding registers	40001–49999	0–9998	Read holding register
Holding registers	40001–49999	0–9998	Preset single register
Holding registers	40001–49999	0–9998	Preset multiple registers
Holding registers	40001–49999	0–9998	Read exception status
Holding registers	40001–49999	0–9998	Loopback diagnostic test

During the message exchange, the specification is defined in the function field and the target device (or the field device/sub-controller) and an action will be performed according to these. In this study, the "field device" keywords are used in-place of the target device as is defined from the MODBUS protocol documentation. The main controller is employed to initiate a request message and response frames will be generated from the field device(s). The main function of the support codes for the MODBUS protocol and their corresponding descriptions are shown in Table 2 in hexadecimal format.

i. **Data Field:** This is a field of random bytes, and its length depends on the function code that is specified in the function field of the MODBUS protocol message frame. During communication, the main controller specifies the function code and the other information that belongs to the data field in the request message that should be performed at the target while the field device responds to the frame according to a request from the main controller [1,6].

ii. **Error Check Field:** This is the last field in the MODBUS message/frame that contains two bytes of information. This is an error checking (or testing) field that employs a cyclic redundancy check (CRC) to compute the numeric value of the message (or frame). The numeric code detects errors and fortuitous changes to the message frame during transmission [1,6].

Table 2. MODBUS Protocol Function Codes and Corresponding Descriptions.

Function Name	Function Code	Description
Read coil or digital output status	01	The field device responds to the logical coil(s) ON/OFF status.
Read digital input status	02	Read discrete inputs from the field device.
Read holding registers	03	Retrieves the contents of the holding register(s) from field device.
Reading input registers	04	Retrieves the contents of input register(s) from the field device.
Force single coil	05	The ON/OFF status of single logic coil is changed from the field device.
Preset single register	06	To change the content of a single holding register.
Read exception status	07	To retrieve the status of eight digital points as a short message request from the field device.
Loopback test	08	Employs diagnostic features including CRC errors and reports according to exceptions to test the operation of the system.
Force multiple coils or digital outputs	0F	To manage the ON/OFF status of the coils (or group of coils).
Force multiple registers	10	To change the content of a single register and to manage a group of coils

The section below presents details related to the MODBUS protocol messaging service over the TCP/IP protocols. The main components have been considered and are explained to understand the communication flow and are to be further deployed and employed in the development section.

3. MODBUS Messaging on TCP/IP

The MODBUS protocol messaging service provides real time communication between field devices that are geographically located at a distance over the Internet. The MODBUS protocol consists of an OSI application layer messaging protocol and base with a client/server architecture model. The interconnectivity between various field devices can be achieved by employing the TCP/IP protocols that provide messaging services over the Internet. Communication is initiated at the main controller by building an application data unit (ADU), and function codes are placed to define the MODBUS messaging meaning and the actions that shall be taken by the target device [1,5–7].

In Figure 4, a header that is referred to as the MODBUS application protocol header (MBAP) is employed to identify the MODBUS ADU while the data is carried over the TCP/IP network, either as a request message or as a response frame. This header creates some functional differences in comparison to the MODBUS serial line application data unit (ADU). A few of these are described as follows [1,5–7]:

- In the MODBUS serial line, a “Unit Identifier” byte is used in-place of the slave address field in the MBAP header, and it is employed to communicate with network devices including routers,

gateways and bridges that are usually configured with a single Internet protocol (IP) address to support several individual MODBUS field devices.

- During transmission, the main controller/sub-controller verifies the content of the messages that were sent, such as a request message and response message. A function code is enough if a fixed-size MODBUS protocol data unit (PDU) has been generated. Otherwise, a one-byte counter is added in the data field in the case where a variable amount of data was carried by the function codes.
- During MODBUS messaging over TCP/IP, additional information is accounted for in the MBAP header that should indicate the target device is full and to identify the multiple packets that have been received during the transmission or in-case the messages have been divided into several packets for the MODBUS TCP/IP transmission. Several implicit/explicit safety rules and computed CRC codes have been employed for the MODBUS messages (such as request and response messages) that result in a minor possibility of unexposed corruption.

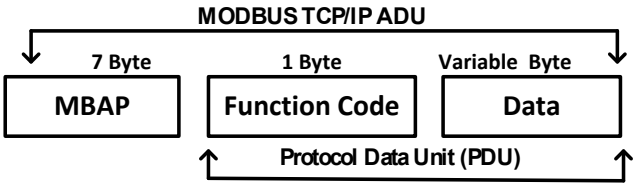

Figure 4. MODBUS Messaging on TCP/IP.

Typically, the size of the MBAP header is seven bytes in length and contains four fields that identify the MODBUS ADU over the TCP/IP protocols [1,5–7]. Figure 5 visualizes the MBAP header field with the occupied bytes.

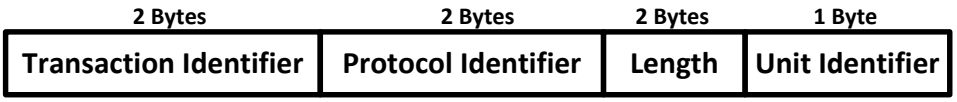

Figure 5. MBAP Header Fields.

i. **Transaction Identifier:** This field is two bytes in length, and is employed for transaction pairing purposes. In the response message, the field device places the transaction identifier of the main controller request.
ii. **Protocol Identifier:** This field contains two bytes and is employed for intra-system multiplexing. A value of "0" is placed to identify the MODBUS message protocol.
iii. **Length:** The length field occupies two bytes, and these are employed to count the bytes of the fields including the unit identifier field and the data field.
iv. **Unit Identifier:** This field contains one byte of information that is employed for intra-system routing between the MODBUS serial line and the TCP/IP networks (via the allocated 502 port). During transmission, the main controller places (or sets) the field value in the request message and the field device (or sub-controller) must transmit the response with the same value set.

Functional Description: MODBUS Architecture Model and TCP/IP Stack

A number of components and corresponding functions have been employed to define the MODBUS communication architecture as well as the connectivity with the TCP/IP stack and other messaging services [1,5]. Figure 6 illustrates the MODBUS messaging architectural model with the corresponding TCP/IP connectivity.

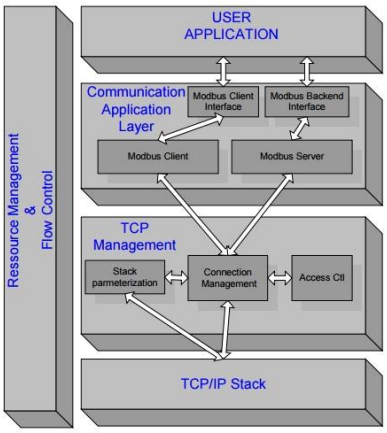

Figure 6. MODBUS Messaging Architecture and TCP/IP Stack [5].

Communication Application Layer

The end user interfaces or main controller/sub-controller interfaces for MODBUS protocol are designed to provide graphical facilities to manipulate messages, such as a request from the main controller and a response from the sub-controller and *vice versa*. The MODBUS backend interface is used to permit indirect access to the user application objects, and the four main areas including the input discrete, output discrete, input registers and output registers that are formulated by this interface and pre-mapping functions are required have to be conducted between the backend interface and the user data [1,5].

i. MODBUS Client and Interface: In this study, we have used a main controller as the client and have achieved an improved MODBUS transmission with a user application that is permitted to control the overall information being exchanged with the target device. The main controller interface is designed to provide and visualizes basic parameters that are required by the user application to generate request messages and to access the MODBUS services and/or its objects.
ii. MODBUS Server and Interface: The field device or sub-controller is designated as a server that generates response messages back to the main controller. Typically, the client sends a request message and the server will reply accordingly, but in a MODBUS transmission the response is transmitted from the sub-controller (or from the target field device). Therefore, due to the client/server specifications, we have replaced certain words with the "client" as a "main controller" and a "server" as "sub-controller".

The MODBUS server interface (or MODBUS backend interface) is designed to communicate with the user application in order to define and manipulate application objects.

TCP Protocol Management Layer

The MODBUS protocol messaging service provides functions to organize and control overall communication, including starting and ending, managing and controlling the byte flow over TCP protocol [1,5].

i. Connection Management: The MODBUS protocol is a messaging protocol that is employed in SCADA systems, and communication between controllers takes place via the TCP protocol. Therefore, a connection management module is required between these controllers. There are two ways in which the TCP connection is managed. One is when the user application itself manages the TCP connection and the other is when the connection is transparent to the user application and is managed by employing a management module.
ii. By default, port 502 is used to listen to the MODBUS protocol communications over the TCP connections. However, some MODBUS applications may require distinct port numbers to listen to TCP traffic. Thus, the best solution is to allow the end-user to configure the TCP ports numbers according to the application specifications and the configuration facility provided by the MODBUS interfaces. In the case where distinct ports are employed and configured by particular applications to access protocol services over the TCP connections. A dedicated TCP port 502 is also available as an additional port number aside from the port(s) specified by the application.
iii. Access Control Module: In a few critical scenarios, the internal, sensitive information of the field devices is accessible for un-authorized recipients. Therefore, a security module is required and needs to be implemented to provide protection against unknown participating hosts.

TCP/IP Stack Layer

The TCP/IP stack is used to parameterize the address, and connection management is used to provide distinct features to address the particular limitations in the MODBUS messaging system specification in order to manage flow control. Berkeley (BSD) sockets are typically used for such purposes [1,5].

Resource Management and Data Flow Control

In the MODBUS protocol messaging stack, a data flow control module is employed to manage incoming and outgoing data (or communication) flow between the main controller and a sub-controller and vice versa. At the start, the internal TCP manages the resource and data flow mechanism, and some additional data flow controls will be added with TCP internal control (or flow control) at the data link layer and user application layer [5].

The following sections discuss security, byte control, and additional security features that are added as part of the cryptography buffer (CB) to improve the existing security of the MODBUS messaging protocol. After security development, the bytes are encapsulated in TCP/IP protocols in order to make it possible for these to travel over the network and/or the Internet.

4. Security Design and Development

In Figure 7, a security development module is added before transmitting the MODBUS PDU to open networks or protocols. Open protocols/networks may suffer from several vulnerabilities during communication over the Internet [15,22–32]. Therefore, the information needs to be kept as secure as possible as part of the normal operation of the system rather than depending on open and unreliable sources [22,48,49]. Figure 7 shows a security module (or security bytes) that are placed and designated as a new MODBUS security development module (NMSDM). The original size of the data is not limited (or fixed), however, the data size should be limited to 253 bytes, plus 7 bytes of the MODBUS application protocol header (MBAP) [1,7,22]. This means that each data unit contains 260 bytes that should be encapsulated in the payload and transported in the form of TCP packets.

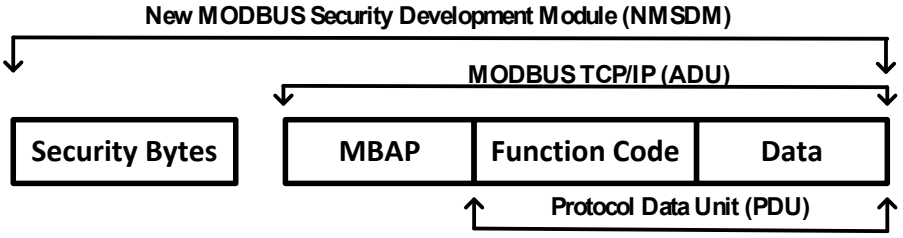

Figure 7. New MODBUS Security Development Module.

Two security developments have been made to secure the MODBUS protocol communications including unicasting and broadcasting. According to the communication requirements, security solutions are designed and deployed on PDU bytes. We assumed that PDU bytes have been secured before, and are insecure after developing security (solutions) [49–53]. The header bytes or the MBAP bytes are not accounted for during security deployment, and encrypted header bytes usually cannot be read at the target side (or remote side) and are also hard to manipulate during transmission. At the moment, we have employed three cryptography algorithms, including AES, RSA and SHA-2, to design and deploy security on the PDU bytes. However, this work is also open to deploy and test other cryptography algorithms, but communication requirements need to take into account the best performance analysis.

For MODBUS message unicasting, three algorithms including AES, RSA and SHA-2 have been employed to verify the security services (or parameters) including authentication, integrity, confidentiality and non-repudiation. Public key cryptography can be used since restricted time independency and transmission are carried out from node-to-node. In this study, however, asymmetric encryption is considered to be inappropriate for broadcasting communication because the numbers of the cryptographic keys (*i.e.*, RSA private and public keys) would be required between the participating nodes in the testbed, and a certificate authority (CA) is required, which is one limitation of this study. The following steps are listed in order to deploy security on the PDU bytes, and proof is subsequently used to validate security.

i. At the beginning, the main controller initiates communication with the sub-controller, and the network nodes are configured and are known in advance, which restricts the unknown transmission entities.

ii. The main controller uses the secret key (that is, one that is generated using the AES algorithm) to encrypt the PDU bytes. This key is also shared with the sub-controller through a secure channel connected between them. The output or encrypted PDU bytes are stored and are designated as M_1. The SHA-2 hashing function is also deployed on the PDU bytes to compute the hash digital of the bytes, and the results are stored and designated as M_2.

iii. The computed hash value (or M_2) is treated with a private key (generated using the RSA algorithm) that forms a digital signature to verify the non-repudiation security parameter. The output or digital signature bytes are stored and are designated as M_3. Subsequently, the target public key (*i.e.*, generated from the RSA algorithm) is employed to encrypt M_1 and M_3.

iv. Upon receiving at the target side, the target private key and the sender public key are used for decryption. The shared secret key is further used to decrypt M_1, and the SHA-2 hashing digest is computed to verify the integrity or to conclude that the contents of the message have not changed during transmission.

For the testbed, each node, such as the main controller and sub-controller, contains private and public key pairs from the RSA algorithm, shared secret key from the AES algorithm and a computed hash digest. During security development, designated cryptography algorithms are deployed, and the results are indicated in short form as M_1, M_2, and M_3.

For critical scenarios (e.g., device authentication, set alarms, device/system normally/abnormally offline and cases where the points are changing), the numbers of times that the main SCADA controller needs to broadcast a message to each and every node in testbed is known, and the network configuration is static, meaning that the number of network nodes is known in advance to avoid entry by new nodes or an unauthorized entity. In our case, the MODBUS message broadcasting cannot use asymmetric encryption (*i.e.*, RSA algorithm), and since a number of communication nodes are configured in the SCADA/MODBUS testbed, the number of public and private keys is also required during transmission. Public key encryption has several benefits, including strong security protection against adversaries, providing digital signature and eradicating key distribution. However, the encryption/decryption processing speed is slow relative to that of symmetric encryption [54,55]. In this study, we do not use the RSA algorithm but rather AES and hashing algorithms are deployed to improve the security of MODBUS messaging during a communication broadcast. However, the non-repudiation security service (or parameter) becomes a limitation of this communication scenario due to the absence of a digital signature.

The secret key is employed to encrypt the PDU bytes and the corresponding result is stipulated as M_4. The hash digest is then computed and designated as M_5 to verify the integrity of the PDU (bytes). The message is then broadcast to the network, and each target node employs a shared secret key and a hash digest to verify the contents of the message. A minimal impact is computed during transmission because each target node employs a shared secret key and a hash function only.

System Model

In this section, the proposed security design and development is validated according to the Postulate 1 (MODBUS Message Unicasting) and Postulate 2 (MODBUS Message Broadcasting). More details now follow:

Postulate 1 (MODBUS Message Unicasting): If there is a one-to-one function $f_\mu: f_\mu(X_s) = f_\mu(X_R)$, then the protocol bytes X are accounted for during security development (Sym^f, Aym^f, H^f).

Preliminary cryptographic keys are generated and distributed in among the participating nodes. In our case, we have used a main controller and a sub-controller. The certificate authority (CA) is a limitation in this study, so keys are generated and distributed statistically over a secure channel established between the controllers. The cryptography keys are: $K_i \rightarrow Sc_{(MC,BC)}$ as the secret key (Sc) shared between the main controller (MC) and the sub-controller (BC) encryption with index *i*, and $K_j \rightarrow (Pr_{(MC,BC)}, Pu_{(MC,BC)})$ as the main controller/sub-controller private (Pr) and public Keys (Pu) with index j. SHA-2 hashing is defined as "H" and the setup as "SP". Two explicit indexes are defined as i and j to indicate security functions such AES and RSA and parameters such as secret keys, private keys and public keys.

$$f_\mu(X_s) = f_\mu(X_R) \Leftrightarrow f_\mu(X_s)$$

$$\Rightarrow \left\{ \exists: \forall \mu_{(M,K,E[K])}: Eny_{\substack{E_\rightarrow Sym^f \\ E_\rightarrow Aym^f, \\ H_\rightarrow H^f}} \left\{ \exists: \forall X^{\alpha=\Delta}. \sum_{f_\rightarrow f_n}^{lim \leftarrow B} \left(f_n: X_{f_\rightarrow f_n}\right) \right\} \right\} \wedge$$

$$f_\mu(X_R) \Rightarrow \exists: \forall \mu_{(M,K,D[K])}: \mathrm{Dny}_{\substack{D_\rightarrow Sym^f \\ D_\rightarrow Aym^f, \\ H_\rightarrow H^f}} \left\{ Eny_{\substack{E_\rightarrow Sym^f \\ E_\rightarrow Aym^f, \\ H_\rightarrow H^f}} \left\{ \exists: \forall X^{\alpha=\Delta}. \sum_{f_\rightarrow f_n}^{lim \leftarrow B} \left(f_n: X_{f_\rightarrow f_n}\right) \right\} \right\} \vee$$

$$\mathrm{Dny}_{\substack{D_\rightarrow Sym^f \\ D_\rightarrow Aym^f, \\ H_\rightarrow H^f}} \Rightarrow$$

X is the number of PDU bytes (B), and f_μ is a function that computes the security for these bytes. The decryption function (Dny) implies on message (M), and (M_1, M_2, M_3) are implied from the encryption function (Eny). Table 3 summarizes the terminology employed in Postulate 1.

Table 3. Unicasting Security Terminologies.

Notations	Description
$f_\mu: f_\mu(X_s) = f_\mu(X_R)$	One-to-one function that satisfied the security.
(Sym^f, Aym^f, H^f)	Symmetric function (Sym^f) , Asymmetric function (Aym^f), and Hashing function (H^f).
K_i	Defines the numbers of secret keys with index *i*.
$Sc_{(MC,BC)}$	Shared secret key (Sc) of the main controller (MC) and sub-controller (BC).
K_j	Define the numbers of private and public keys with index of j.
$(Pr_{(MC,BC)}, Pu_{(MC,BC)})$	Distinct private key (Pr) and public key (Pu) of main controller (MC) and sub-controller (BC).
$\alpha = \Delta$	User defined pointer. Pointer that indicates the security function and its parameters followed by their index.
Eny/Dny	Encryption (Eny) and decryption (Dny).
M_1, M_2, M_3	User defined short forms indications.

In existing survey [21,27,29,56–58], number of potential attacks, such as Bytes Injection, Man in the Middle attack, Intercepted Information Replay, Abnormal Bytes Transferring, Brute Force Attack, Bytes Injection and Deletion, Shared Key Guessing Attack, Eavesdropping Attack, Key Cracking Attack, and others, were accounted that suffer the SCADA communication, as well as, a taxonomy of the MODBUS attacks has been conducted, and attacks are categorized into four main parts, including bytes interception, bytes interruption, and data modification [22]. As a consequence, we analyzed the potential attacks that are to be considered most harmful for SCADA/MODBUS communications in form of four main parameters, such as integrity, authentication, confidentiality, and non-repudiation. These security parameters are considered as a whole; rather than thoroughly explaining of each attack from each security parameter, but this limitation would be considered as future work.

The RSA algorithm provides strong authentication and confidentiality security services for sensitive information of a system (e.g., SCADA/MODBUS system). In-case, an attacker is residing in between the main controller and sub-controller and/or vice versa, and wants to stolen the sensitive information of SCADA/MODBUS communication by using attack tools (as described in Section 6) or by using other methods [27,53,59,60], the security development (Sym^f, Aym^f, H^f) is required to decrypt the encrypted Message (M), and (M_1, M_2, M_3), which is not easy for attacker, as well as, he/she also required depth knowledge of MODBUS protocol.

Postulate 2 (MODBUS Message Broadcasting): The function f_{BT}: $f_{BT}(\mathrm{BS}_s)$ and function f'_{BT}: $f'_{BT}\{f_{BT}(\mathrm{BS}_s)\}$ are broadcasting functions, if and only if parameters (SS, GK, Eny, Dny) are satisfied. The mapping function f_{MP}: $\mathrm{BS}_s \longrightarrow \mathrm{BS}_R(\mathrm{G}_i) \implies$

$$\mathrm{BS}_s \longrightarrow f_{BT} \longrightarrow f_{BT}(\mathrm{BS}_s) \longrightarrow f'_{BT} \longrightarrow f'_{BT}\{f_{BT}(\mathrm{BS}_s)\} \longrightarrow \mathrm{BS}_s \in \mathrm{BS}_R(\mathrm{G}_i)$$

$\implies SS(i, \Delta, G, KC), G \leq i$, $KC \longrightarrow (Sc_{(MC,BC)}$, H) represents the system setup (SS), with i as the number of nodes participating in the broadcast group (G) such that $G \leq i$, and a key counter (KC) with index pointer Δ.

$\implies$ GK $(l, \Delta, Sc, \mathrm{H})$ is the number of keys (Sc, H) employed in sequence such that, $l = l_0, l_1, l_2, \ldots\ldots, l_{limit-1}, l_{limit}$ with index pointer Δ where GK stands for the key group and is employed to manage keys.

$\implies$ Eny(BS, Δ, Sym^f, H^f), the number of bytes or set of bytes (BS) is computed with security functions (Sym^f, H^f) and index pointer Δ. The bytes are variable and then $BS \leq limit$. The results of the encryption computation are designated as $payload_1$.

$\implies$ Dny$(payload_1, i, G, \Delta, Sym^f, H^f)$, each target node i in group 'G' G is able to receive the $payload_1$ for the decryption (Dny). Security functions (Sym^f, H^f) are also employed with index pointer Δ. The decryption of the $payload_1$ is used to recreate the original $payload$. Table 4 summarizes the terminology that is employed in Postulate 2.

As consequence, symmetric encryption with the AES algorithm and hashing using the SHA-2 algorithm are deployed in order to secure the PDU bytes while the message is broadcast from the main controller to the sub-controllers in the testbed. The AES and SHA-2 algorithms provide authentication, confidentiality and integrity security services and protection from corresponding attacks during the broadcast transmission. In-case, there is an attacker that wants to launch the attacks (as described in Section 6) in between the broadcasting transmission while $payload_1$ has been broadcast to group (G). However, it is not easy to decrypt the $payload_1$ to original $payload$ due to strong security development

(of AES and SHA-2 algorithms), with the complicated knowledge of MODBUS protocol as a part of SCADA system.

Asymmetric encryption (*i.e*., the RSA algorithm) is considered to be a better security approach than symmetric encryption (*i.e*., AES algorithm) and it also provides strong security against authentication and confidentiality attacks for SCADA/MODBUS transmissions. However, this study does not employ the RSA algorithm in the MODBUS broadcast mode due to the number of keys that need to be generate and need to be managed, so this limitation should be considered for future security development of the MODBUS broadcasting transmission. We applied and tested the RSA algorithm for MODBUS unicasting transmission where the PDU hashing value is computed, and the private RSA key is deployed on this hashing value to produce a digital signature of the PDU bytes in order to prevent non-repudiation attacks [22,27,60–66].

Table 4. Broadcasting Security Terminologies.

Notations	Description
$f_{BT}: f_{BT}(\mathrm{BS}_s)$	Broadcasting (BT) function (*f*) computed on the set of bytes (BS) at sender(s) side.
$f'_{BT}: f'_{BT}\{f_{BT}(\mathrm{BS}_s)\}$	Broadcasting (BT) function (*f*) computed on the set of bytes (BS) at target side.
f_{MP}	Mapping(MP) function(*f*)
$\mathrm{BS}_R(\mathrm{G}_i)$	Set of bytes (BS) received by each node (*i*) in group (G).
$SS(i, \Delta, G, KC)$,	System setup (SS) with parameters such as nodes (*i*), index pointer (Δ), group (G) and key counter (KC).
$(Sc_{(S,i)}, \mathrm{H})$	Shared secret key of the sender (S), and each node (*i*) and hashing (H).
(Sym^f, H^f)	Symmetric function (Sym^f) and hashing function (H^f).
Eny/Dny	Encryption (Eny) and decryption(Dny)
$payload$	Original protocol bytes.
$payload_1$	Security bytes computed at the sender side by employing a symmetric function (Sym^f) and a hashing function (H^f).
Δ	Pointer that indicates the security function and its parameters, followed by their index.

5. Cryptography Buffer

The proposed study implemented a cryptography-based security solution in the MODBUS messaging protocol before transmission to open protocols, such as TCP/IP protocols (over the Internet). In order to manage and control the overall security development, as well as to monitor the MODBUS messaging protocol and transmission, a cryptography buffer (CB) is proposed [67]. The CB contains variable bytes, and bytes are come in and out from the buffer according to the given requirements. However, according to our best analysis, the size of the buffer is considered to be of 30 bytes. In the MODBUS PDU, the size of the data is not limited, so we can employ 30 bytes for the data field. For example, if 200 bytes are defined in the data field, then we consider 270 bytes by including the CB buffer. For few cases, if the size of the data field is fixed [1,5,22], then the CB also uses the bytes from the data field by fixing the bytes.

For example, a data unit = 253 bytes and MBAP = 7 bytes.

Then, we reduce the data unit size to 230 and the MBAP bytes remain the same. However, the TCP protocol carries 260 bytes of MODBUS ADU as payload in its packet each time. Why do we use the data unit (data field) bytes? We need to consider CB as part of the protocol and not an external

development. The cryptography buffer (CB) has been integrated at both sides and employs a number of fields with occupied bytes of variable lengths, as illustrated in Figure 8.

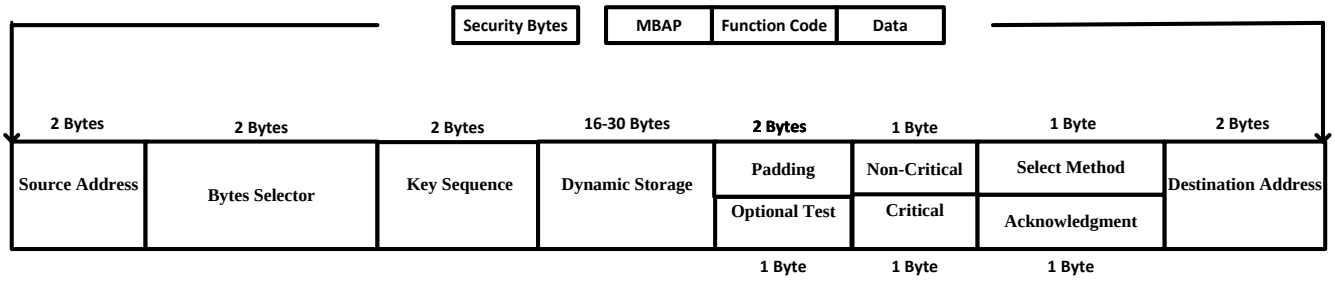

Figure 8. Cryptography Buffer (CB) Fields.

i. Byte Selector: The MODBUS protocol PDU has been constructed (or is ready) and becomes ready to transmit. The CB keeps track of these bytes as well as the security development bytes in its functional field, which are referred to as "Bytes Selectors". This defines how many bytes are constructed and proceed to the TCP protocol. As explained above, the TCP protocol carries 260 bytes of the MODBUS ADU as a payload in its packet.

Socket = listenConnectionTCP(Port 502);
Transmit (Socket, data);
Close (Socket);

ii. Key Sequence: The cryptographic keys are employed with unique sequences of numbers and are added in the key sequence counter. This counter keeps the track of the keys that are used or are being used by the main controller/sub-controller(s).

iii. Padding: This field initially occupied two bytes of information and is deployed to define how the MODBUS message is constructed, and the remaining bytes are padded with zeros.

iv. Optional Test: A byte function field that is usually deployed at the end to verify the contents of the message before transmission to protocols/networks.

v. Critical and Non-Critical: Two byte fields that keep the information for abnormal and normal MODBUS communication. A short security code travels along a message that keeps information of the communication. This type of code is frequently employed when transmission occurs for a number of times that follow specific intervals.

vi. Select Method: One byte of information is kept by this field and is used to change the security development according to the communications requirements, such as unicasting and broadcasting communications.

vii. Acknowledgment: Here, an acknowledgment is treated as an exceptional message that is generated locally during development, such as when security is successfully implemented, and additional bytes are required from the dynamic storage, and security method is needed to change and for other purposes.

viii. Source and Destination Addresses: In a few cases, the target node(s) are not able to read the encrypted information and the encrypted header bytes from the sender. Thus, an external header is transmitted to the target node along with the encrypted bytes (or message). These are fields with four bytes in length and are usually employed in the case of the MODBUS serial messaging.

ix. Dynamic Storage: This field contains 16–30 bytes of information and is considered to be a special case for the CB. As the name implies, the bytes are dynamically allocated to other fields of the CB according to demand. In potential attack scenarios, the attacks are not able to steal sensitive information of the MODBUS protocol due to the encryption, but information can be stolen from the header (or MBAP header). In this case, the MBAP header is also encrypted and an identical copy of the MBAP header is transmitted to the TCP protocol. Therefore, seven bytes are used from dynamic storage in order to provide an identical copy of the MBAP header. This is therefore a good approach to keep the header information secure from attackers, and another solution involves the use of a hashing algorithm. For example, the hash value of the header is computed and should be verified at the target side.

6. Testbed Setup and Measurement

A SCADA/MODBUS testbed (as illustrated in Figure 9) was designed to contain a specific number of nodes or sub-controllers (e.g., 16 nodes) configured with main controller, the messaging specification of the main controller and each node are followed by the MODBUS TCP/IP protocol. In other words, a new MODBUS security development module (NMSDM) has been installed in each node, including the main controller. The network configuration of the testbed is straightforward where sixteen nodes are connected with the main controller through three routers, such as Router 1, Router 2 and Router 3, Nodes 1–8 are connected with Switch 1 in Station 1, and Nodes 9–16 are connected with Switch 2 in Station 2. The unit identifier is employed in-place of the slave address field in the MBAP header and is employed to communicate with network devices, such as the switches and routers. However, networks paths are defined in advance to restrict unknown entities during transmission. If any new node needs to be added, it must first be registered at the main controller side (or in the routing group).

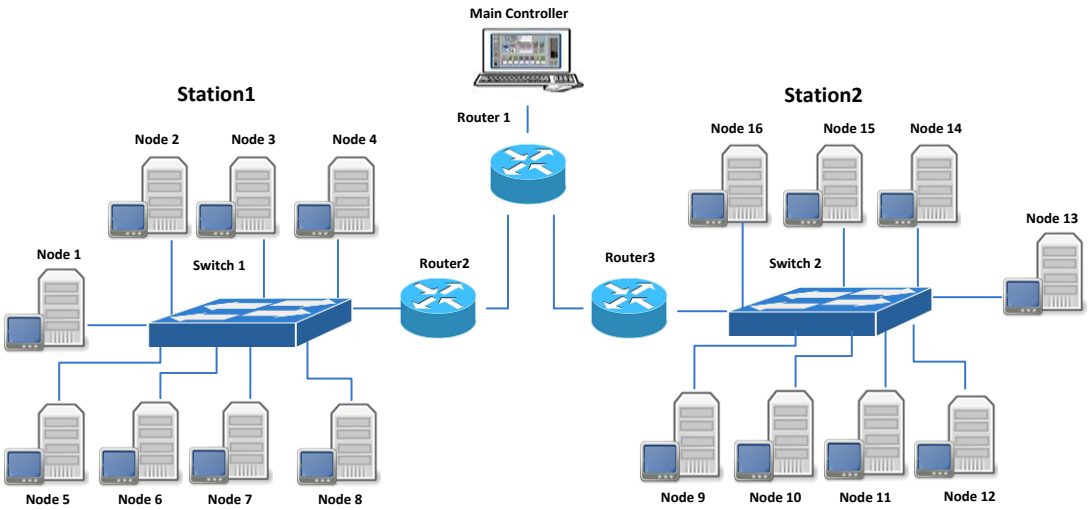

Figure 9. Testbed Setup and Configuration.

The number of times MODBUS messages are transmitted between the participating nodes in the testbed and security is tested to validate developments. The messages are then computed with distinct MODBUS function codes, and security is implemented, encapsulated in TCP packets and transmitted to the network. On the other side, security is computed and verified to show an accurate transmission flow. The examples

below are considered to define the MODBUS message function codes, security bytes in the request message and the corresponding security at the target side would be computed for verification.

Example 1: ≪ 0x01, 0x1E, 0x2E, 0x3E ≫ ≪ 0x01, 0x1D, 0x2D, 0x3D ≫

Example 2: ≪ 0x02,0x1E, 0x2E, 0x3E ≫ ≪ 0x01, 0x1D, 0x2D, 0x3D ≫

In the above examples, the function codes 0x01 and 0x02 are employed to read coils and read discrete input. Codes 0x1E, 0x2E, 0x3E are security development codes at the main controller, and codes 0x1D, 0x2D, 0x3D are computed at the target side or are designated for decryption. The defined security codes are logical. However, we can verify the MODBUS TCP/IP communication flows by using [68], and these are visualized in Figure 10:

```
Command Window repeat interval initiated "Read
Input Registers" at 23:21:12
>>> MainMBreadiregs session 0 start 0 quantity 3
23:21:12.358: <+++ MainMB      Insert request in
queue: Read Input Registers
Command Window repeat interval initiated
"Read/Write Holding Registers" at 23:21:12
>>> MainMBreadwritehregs session 0 start 0 quantity
2 value 0
23:21:12.358: <+++ MainMB      Insert request in
queue: Read/Write Multiple Registers
23:21:12.421: ===> SubMB      Application Header,
Read Discrete Inputs
23:21:12.421:           Starting Register=0,
Quantity=3
23:21:12.421:          02 00 00 00 03
23:21:12.421:              Discrete Input 0 = 0
23:21:12.421:              Discrete Input 1 = 0
23:21:12.421:              Discrete Input 2 = 0
23:21:12.421: <=== SubMB      Application Header,
Read Discrete Inputs
23:21:12.421:           Byte Count=1
23:21:12.421:          02 01 00
23:21:12.530: ===> MainMB      Application Header,
Read Discrete Inputs
23:21:12.530:           Byte Count=1
23:21:12.530:          02 01 00
23:21:12.530:              Discrete Input 0 = 0
23:21:12.530:              Discrete Input 1 = 0
23:21:12.530:              Discrete Input 2 = 0
23:21:12.530: <=== MainMB      Application Header,
Read Input Registers
23:21:12.530:           Starting Register=0,
Quantity=3
23:21:12.530:          04 00 00 00 03
23:21:12.530: ===> SubMB      Application Header,
Read Input Registers
23:21:12.530:           Starting Register=0,
Quantity=3
23:21:12.530:          04 00 00 00 03
23:21:12.530:              Input Register 000000 =
0x0
23:21:12.530:              Input Register 000001 =
0x0
23:21:12.530:              Input Register 000002 =
0x0
23:21:12.530: <=== SubMB      Application Header,
Read Input Registers
23:21:12.530:           Byte Count=6
23:21:12.530:          04 06 00 00 00 00 00 00
23:21:12.639: ===> MainMB      Application Header,
Read Input Registers
```

```
Command Window repeat interval initiated "Read
Discrete Inputs" at 23:21:17
>>> MainMBreaddinputs session 0 start 0 quantity 3
23:21:17.350: <+++ MainMB      Insert request in
queue: Read Discrete Inputs
23:21:17.350: <=== MainMB      Application Header,
Read Discrete Inputs
23:21:17.350:           Starting Register=0, Quantity=3
23:21:17.350:          02 00 00 00 03
23:21:17.444: ===> SubMB      Application Header,
Read Discrete Inputs
23:21:17.444:           Starting Register=0, Quantity=3
23:21:17.444:          02 00 00 00 03
23:21:17.444:              Discrete Input 0 = 0
23:21:17.444:              Discrete Input 1 = 0
23:21:17.444:              Discrete Input 2 = 0
23:21:17.444: <=== SubMB      Application Header,
Read Discrete Inputs
23:21:17.444:           Byte Count=1
23:21:17.444:          02 01 00
23:21:17.553: ===> MainMB      Application Header,
Read Discrete Inputs
23:21:17.553:           Byte Count=1
23:21:17.553:          02 01 00
23:21:17.553:              Discrete Input 0 = 0
23:21:17.553:              Discrete Input 1 = 0
23:21:17.553:              Discrete Input 2 = 0
Command Window repeat interval initiated "Read
Discrete Inputs" at 23:21:22
>>> MainMBreaddinputs session 0 start 0 quantity 3
23:21:22.358: <+++ MainMB      Insert request in
queue: Read Discrete Inputs
23:21:22.358: <=== MainMB      Application Header,
Read Discrete Inputs
23:21:22.358:           Starting Register=0, Quantity=3
23:21:22.358:          02 00 00 00 03
Command Window repeat interval initiated "Read Input
Registers" at 23:21:22
>>> MainMBreadiregs session 0 start 0 quantity 3
23:21:22.358: <+++ MainMB      Insert request in
queue: Read Input Registers
23:21:22.467: ===> SubMB      Application Header,
Read Discrete Inputs
23:21:22.467:           Starting Register=0, Quantity=3
23:21:22.467:          02 00 00 00 03
23:21:22.467:              Discrete Input 0 = 0
23:21:22.467:              Discrete Input 1 = 0
23:21:22.467:              Discrete Input 2 = 0
23:21:22.467: <=== SubMB      Application Header,
Read Discrete Inputs
23:21:22.467:           Byte Count=1
23:21:22.467:          02 01 00
```

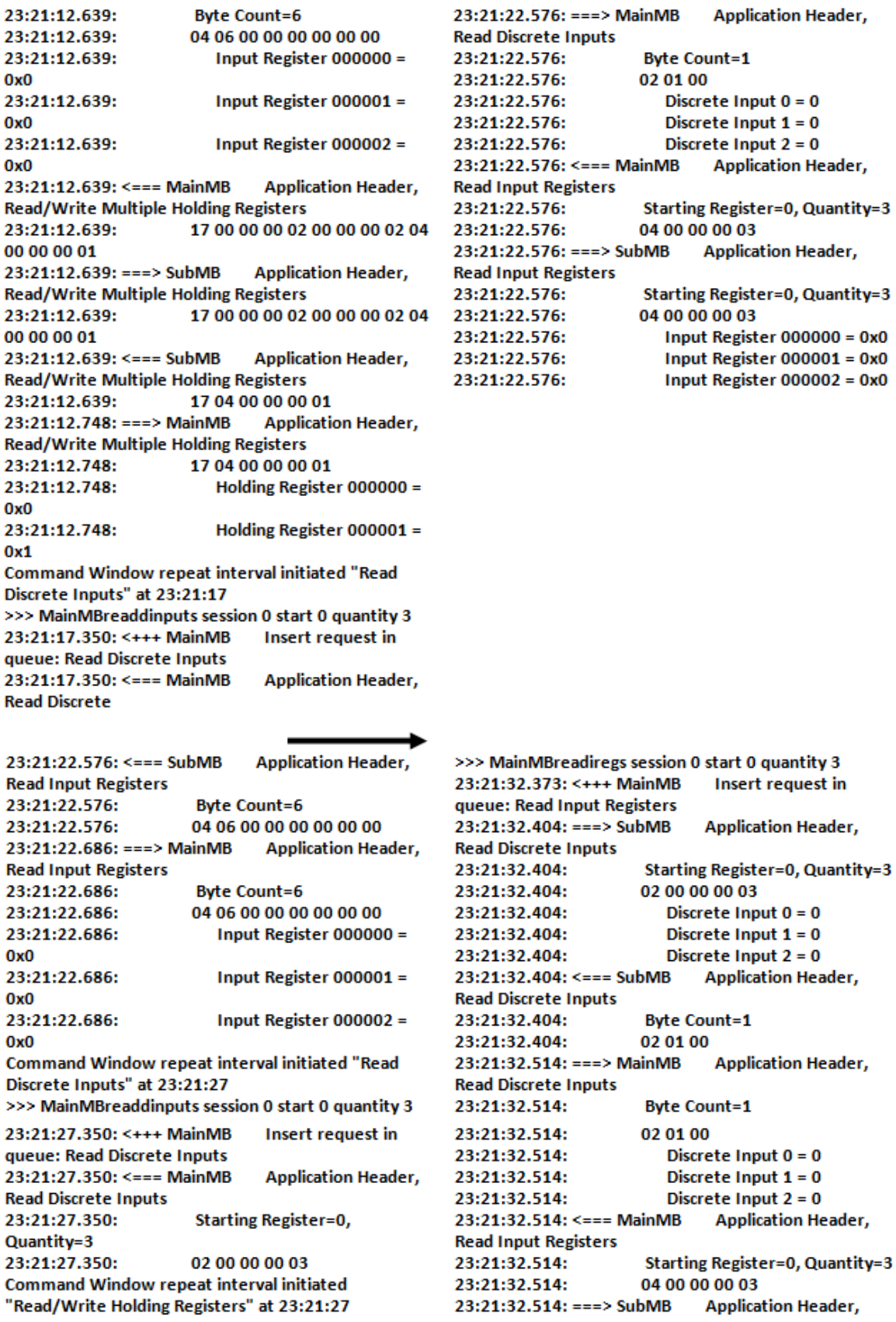

Figure 10. Transmission Flows.

The purpose of the testbed is to conduct attack scenarios during communications, such as unicasting and broadcasting, and to measure the performance of the attack detection and the corresponding security evaluation. In Figure 11, the number of attack types and the corresponding attack tools are illustrated to establish abnormal scenarios in SCADA/MODBUS communications [22,25,69–81]. To validate the security development and its parameters, such as authentication, integrity, confidentiality and non-repudiation (in-case of unicasting communication), the corresponding attacks are launched and the system behavior is observed. For abnormal scenarios, wireless attack tools, such AirCrack, AirSnort, airpwn, and file2air, are also employed in the MODBUS wireless local area network (MODBUS Wireless Local Area Network (WLAN)) [71–73,76,77,82–87], and this additional MODBUS testbed scenario (or wireless scenario) is conducted to examine the proposed security development and to evaluate the corresponding performance during the transmission of sensitive information of MODBUS protocol in the WLAN.

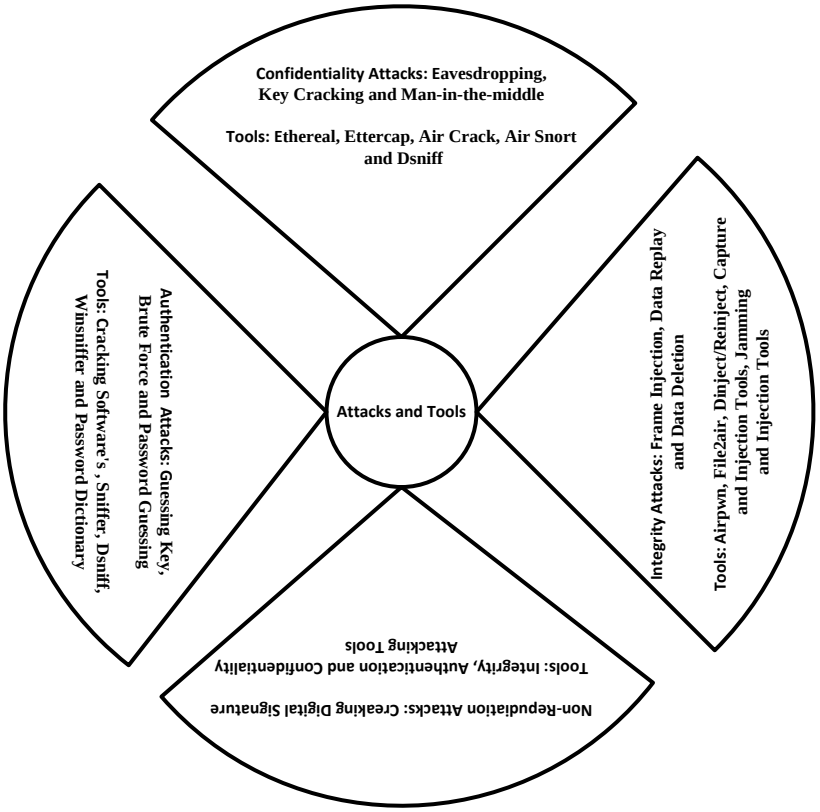

Figure 11. Attacks and Attacking Tools.

In the testbed, attacks are launched 100 times to verify each of the security services (or parameters) as potential attack experiments. However, 50 experiments are sampled and are considered to be the most appropriate, meaning that we considered 50 successful experiments according to the best of our knowledge.

In Figures 12–15, attacks are launched 50 times to verify the proposed security scheme for MODBUS unicast communication. The results indicate that one authentication attack and three confidentiality attacks were detected, and the remaining attacks (such as integrity attacks and non-repudiation attacks) did not interrupt communications. The total percentage of the attack detection in Figures 11–14 is 2%, and the corresponding security is computed at 98%. The calculated measurements are visualized in Figure 16.

A similar attack phenomenon is repeated for the MODBUS broadcast communication. Figures 17–19 show the numbers of attacks that have been detected during abnormal MODBUS broadcast communication. The results of 150 experiments show that two authentication attacks were detected during experiment numbers 7 and 24, four integrity attacks were successfully detected during experiment numbers 8, 24, 40 and 47, and two confidentiality attacks were detected during experiment numbers 11 and 41. Based on the attack detection shown in Figures 17–19, the total attack detection percentage is considered to be 5% and the corresponding security is of 95%, as illustrated in Figure 20. However, non-repudiation service is not a part of MODBUS broadcasting communication.

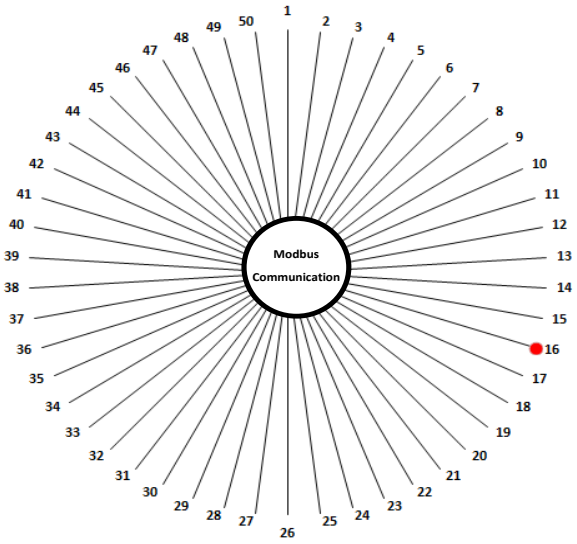

Figure 12. Unicasting: Authentication Attacks.

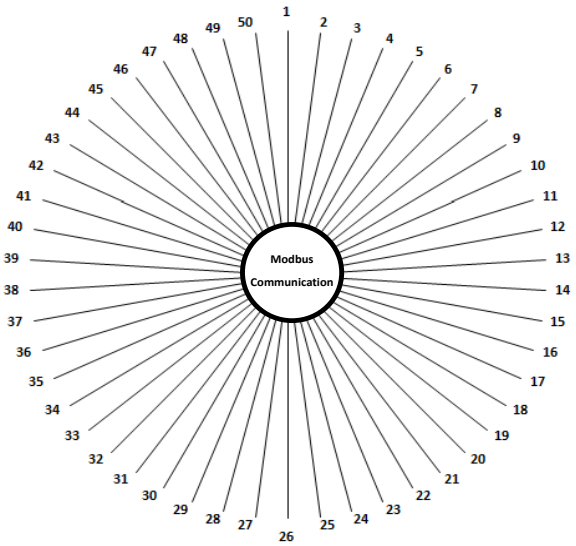

Figure 13. Unicasting: Integrity Attacks.

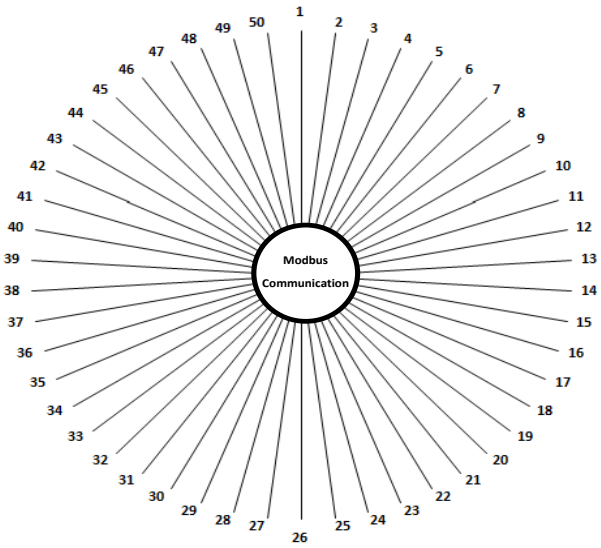

Figure 14. Unicasting: Non-Repudiation Attacks.

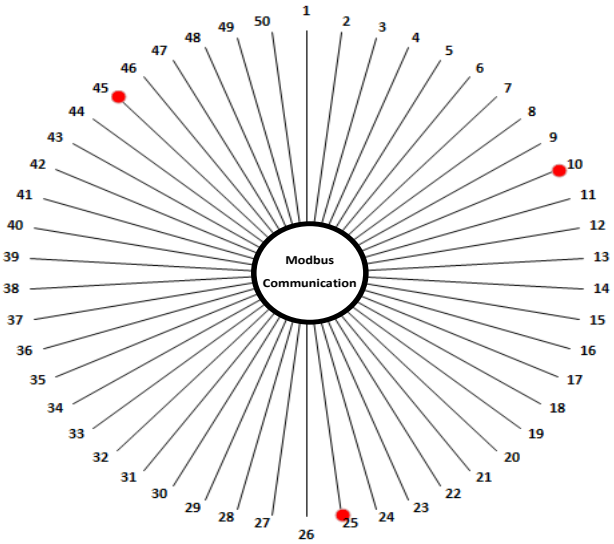

Figure 15. Unicasting: Confidentiality Attacks.

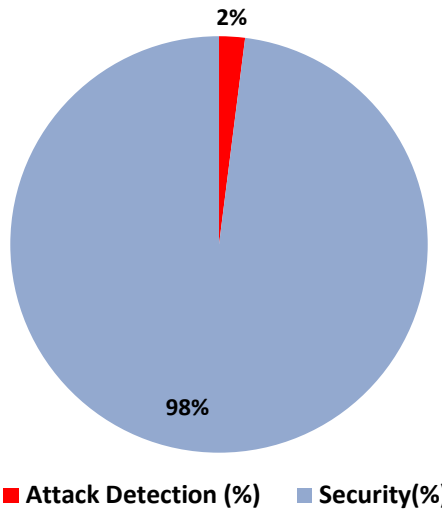

Figure 16. Unicasting: Performance Comparison.

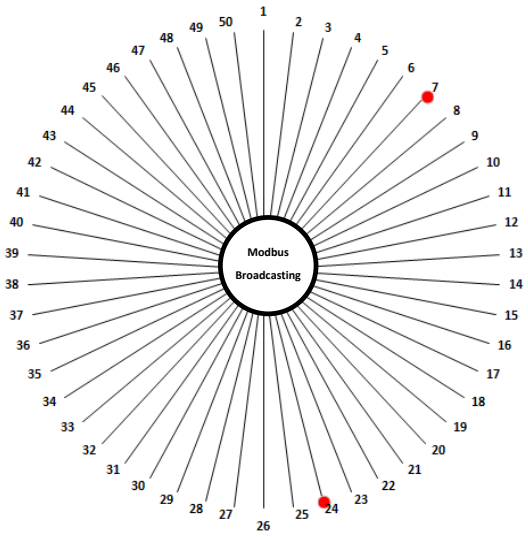

Figure 17. Broadcasting: Authentication Attacks.

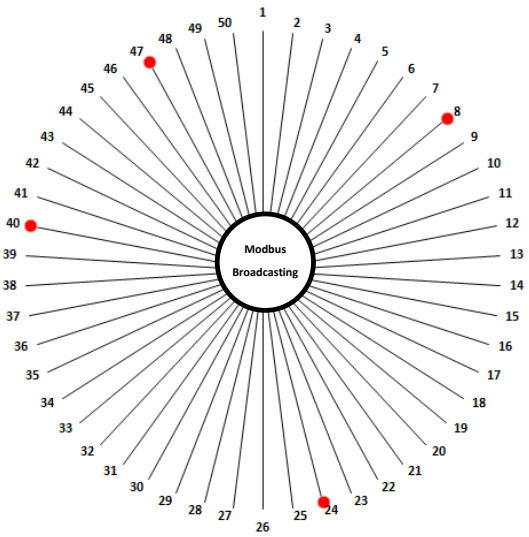

Figure 18. Broadcasting: Integrity Attacks.

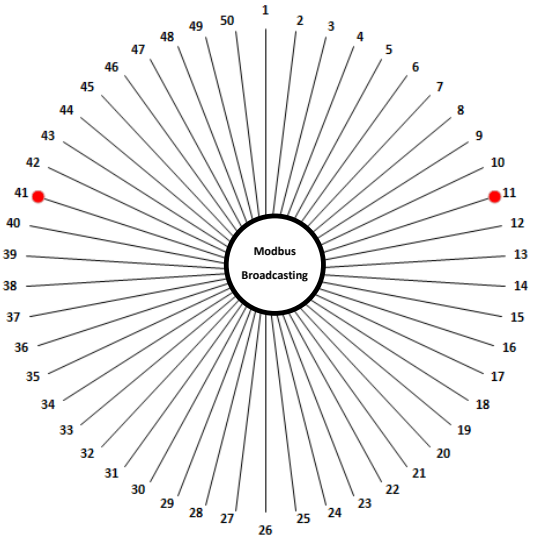

Figure 19. Broadcasting: Confidentiality Attacks.

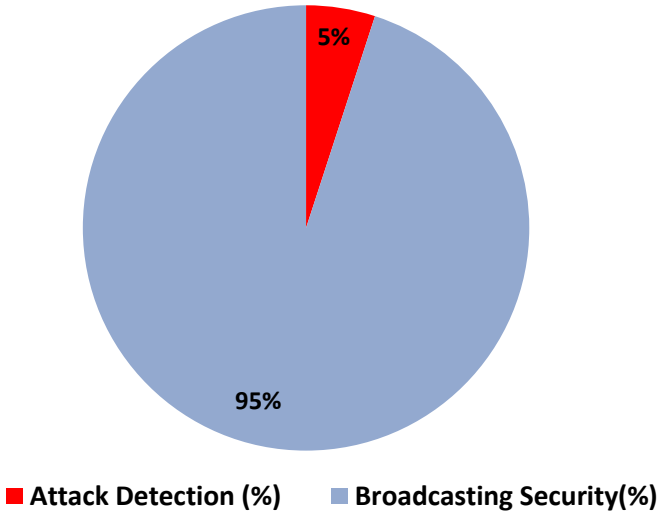

Figure 20. Broadcasting: Performance Comparison.

From the above, we can conclude that a significant level of security has been observed in MODBUS communications, even for abnormal or attack scenarios. To examine the results from Figures 16 and 20, the number of times attacks are launched in the absence of security development is shown [25,51,88]. The computed measurements indicate 95% attack detection and a corresponding security of 5%, which are quite far from the results of this study. More comparisons of the performance are described in the section below.

7. Performance Evaluation and Comparison

This study provides a comparative account of a new security development for SCADA/MODBUS security and discusses its future development. The DNP3 user group has been trying to improve the security of the DNP3 protocol, but most of the work is in the development stage, the initial design of the DNP3 protocol was also without security concerns, similar to the MODBUS protocol [22,50,56,57,89]. Another security improvement has been made in the DNP3 protocol by reference [58] where the cyclic redundancy check (CRC) bytes were used and security development was taken into account to improve the security of the SCADA/DNP3 security. New functional fields were placed in the original DNP3 protocol frame. These fields are: a new security header, a sequence number and authentication bytes (or data). The session key mechanism is used to secure communication over the SCADA/DNP3 protocol, and the security computations are limited to an authentication and encryption. However, these methods are manipulated separately to verify the security services, such as authentication and confidentiality of the data [90,91]. The analysis produced results that emphasized general security computations, without any formal security deployments, and examined proof and testbeds.

The analysis above allows us to conclude that security is important and needs to be deployed in the SCADA protocols rather than with end-to-end security tests and/or through commercial security software. The proposed research was conducted over a simulated environment that includes the MODBUS protocol model design and a new model deployment, and the measurements were taken into account and were examined to be the most approximate to the best of our knowledge.

Four main security parameters, including authentication, integrity, confidentiality, and non-repudiation, were considered to improve the security of the SCADA/MODBUS protocol and its communication. Cryptography algorithms were successfully deployed to compute the desired security and to verify the

parameter as a whole. This means that each security attack from each security parameter was not thoroughly explained and needs to be considered as future work. However, the proposed security design is suitable for deploying and testing other cryptographic algorithms and it could be available as an open design for end users.

In order to measure the security performance, a testbed is established (in Section 6) and attack scenarios were designed to disrupt the normal flow of communication, such as during MODBUS unicasting and MODBUS broadcasting. The performance computations were straight forward, and when approximate values are taken into account, the numbers of corresponding attacks for each security service (or parameter) are launched, and the potential influence of the attacks is observed. The security percentages for each communication were calculated according to the detection percentage of the attacks. As a consequence, the computed security of 98 percent (in-case of the MODBUS unicasting communication) and 95 percent (in the case of the MODBUS broadcasting communication) were comparatively high and inclusive in terms of the design and development in comparison to references [51,58,88,90,91]. Existing work was not conducted on the basis of producing the security computational percentages or following our inclusive scenarios, and few limitations are observed. A comparison is made in Table 5 on the basis of these limitations. However, the reference [88] conducted a study without deploying a cryptography buffer (or with a conceptual deployment) and transmission is limited to unicasting (or unicasting transmission).

Table 5. Performance Comparison.

Comparison	Security Design/Security Development	Security Test	Testbed/Attack Scenarios	Proof of Security	Transmission
Research [90]	Real/Real	Authentication, Integrity, Confidentiality	Non Existent/Non Existent	Exists	Unicasting
Research [58]	Conceptual/Conceptual	Authentication, Integrity, Confidentiality	Non Existent/Non Existent	Conceptual	Unicasting
Research [51]	Real/Real	Authentication, Integrity, Confidentiality, Non-Repudiation	Exists/Exists	Exists	Unicasting
Research [27]	Conceptual/Conceptual	Authentication, Integrity, Confidentiality	Non Existent/Exists	Conceptual	Unicasting
Proposed Research	Real/Real	Authentication, Integrity, Confidentiality, Non-Repudiation	Exists/Exists	Exists	Unicasting, Broadcasting

In general, several existing studies [91–98] were conducted to deploy and improve security in SCADA systems using cryptography mechanisms. Most of these are based on client/server architecture as part of a SCADA communication system. Normally, the main controller is superior and performs supervisory control over the entire configured network (or SCADA network). The main controller

initiates and sends a request message to field devices that are configured for the physical world, and these are authorized to generate and respond to the main controller according to a request. During transmission, the message is treated with security methods (or cryptography algorithms) to keep the message (or payload) secure from un-authorized users (or attackers) [23–32,59,99,100]. However, these security developments [25,29–31,35–40] are usually end-to-end based, meaning that security developments are not part of a SCADA system and its protocols, but rather that security is treated and computed as part of external component sources. The computed results are far away from our computed results. In conclusion, the approximate security lies in the range of 60 to 80 percent, which is comparatively low relative to the current research, which computed security results of 95 to 98 percent. Therefore, this study and others [25,29–31,35–40,59,91–98,101,102] conclude that security in any system should be significantly improved if a security mechanism can be a part of a system rather depending on an end-to-end approach. However, security developments are quite difficult to design and test inside of the protocol and the required depth knowledge of the protocol and other implementation details except security performance are more accurate and remarkable in these scenarios.

8. Significance of Research

There are many studies [22,25,29–31,35–40,91–98] that have been deployed to secure communication for MODBUS protocol communication as well as SCADA/protocol communication. Most of these provide adequate security computation and performance, but cryptography based mechanisms can be considered to be the best approaches for computer networks (such as traditional, and real time networks). Therefore, the details of the security analysis indicate that cryptography-based mechanisms can provide security for the SCADA/MODBUS protocol. However, existing developments [22,35–40,69,91–98,103–109] have not taken into account SCADA/MODBUS communications due to end-to-end limitations. Therefore, the proposed research identifies a new development in place of an end-to-end scheme. An assessment of the results can be used to conclude an improvement was made for the MODBUS protocol communication, and the proposed work is significant in terms of future potential for development as well as for other SCADA protocols, such as DNP3, Fieldbus, Profibus and International Electrotechnical Commission (IEC) 60870-5-104.

9. Conclusions

In this study, we produced an efficient, inclusive security development that not only provided security for communication in the MODBUS protocol but also provides immense trends for future development and improvement of SCADA systems. We have selected, deployed, and tested the most prominent cryptography algorithms to conclude that the platform provides remarkable performance. However, this platform is also open to deploy and test other security algorithms in the area of cryptography. Therefore, according to the transmission demands, end-users are able to deploy security in the MODBUS protocol as a part of the SCADA system. For proof, formal proofs were generated, and the testbed setup was created to carry out attack scenarios that were designed. The corresponding performance was computed to substantiate the proposed scheme, and the performance was examined. However, cryptography keys were distributed in the absence of a certificate authority (CA), which will be considered as part of future work [102].

Acknowledgment

This research was supported by the Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (NRF-2014M3C4A7030503).

Author Contributions

In this research, Aamir Shahzad, Young-Keun Lee and Malrey Lee conceived and designed the experiments; Aamir Shahzad and Suntae Kim performed the experiments; Aamir Shahzad and Naixue Xiong analyzed the data; Aamir Shahzad, Jae-Young Choi and Younghwa Cho contributed materials/analysis tools; Aamir Shahzad, Young-Keun Lee and Malrey Lee wrote the paper.

Conflicts of Interest

The authors declare no conflict of interest.

References

1. Reynders, D.; Mackay, S.; Wright, E. *Practical Industrial Data Communications: Best Practice Techniques*; Butterworth-Heinemann: Oxford, UK, 2004; pp. 132–147.
2. Stouffer, J.; Kent, K. *Guide to Supervisory Control and Data Acquisition (SCADA) and Industrial Control Systems Security, Recommendations of the National Institute of Standards and Technology*; NIST: Gaithersburg, MD, USA, 2011; pp. 2–13.
3. National Communications System. *Supervisory Control and Data Acquisition (SCADA) Systems*; Technical Information Bulletin 04-1; National Communications System: Arlington, TX, USA, 2004; pp. 8–12
4. Boyer, S.A. *Scada: Supervisory Control and Data Acquisition*; Instrumentation, Systems and Automation Society: Research Triangle Park, NC, USA, 2004.
5. The Modbus Organization. *MODBUS Messaging on TCP/IP Implementation Guide V1.0a*; Modbus Organization: Hopkinton, MA, USA, 2004; pp. 2–15.
6. The Modbus Organization. *MODBUS Application Protocol Specification V1.1a*; Modbus Organization: Hopkinton, MA, USA, 2004; pp. 2–11.
7. The Modbus Organization. *MODBUS Protocol*; Modbus Organization: Hopkinton, MA, USA, 2000; pp. 2–74
8. Susanto, I.; Jackson, R.; Paul, D.L. Industrial Process Control System Security. *Wiley Handbook of Science and Technology for Homeland Security*; John Wiley & Sons: Hoboken, NJ, USA, 2009; pp. 1–15.
9. Mahmood, A.N.; Leckie, C.; Hu, J.; Tari, Z.; Atiquzzaman, M. Network traffic analysis and SCADA security. In *Handbook on Information and Communication Security*; Stavroulakis, P., Stamp, M., Eds.; Springer; New York, NY, USA, 2010; pp. 383–405.
10. Cai, B.; Liu, Y.; Liu, Z.; Wang, F.; Tian, X.; Zhang, Y. Development of an automatic subsea Blow out preventer stack control system using PLC based SCADA. *ISA Trans.* **2012**, *51*, 198–207.

11. Ozdemir, E.; Karacor, M. Mobile phone based SCADA for industrial automation. *ISA Trans*. **2006**, *45*, 67–75.
12. Edmonds, J.; Papa, M.; Shenoi, S. Security Analysis of Multilayer SCADA Protocols. In *Critical Infrastructure Protection*, IFIP International Federation for Information Processing; Springer US: New York, NY, USA, 2008; Volume 253, pp. 205–221.
13. Digi. Remote Cellular TCP/IP Access to MODBUS Ethernet and Serial Devices. Available online: http://ftp1.digi.com/support/documentation/90000772_a.pdf (accessed on 1 July 2015).
14. Cagalaban, G.A.; So, Y.; Kim, S. SCADA Network Insecurity:Securing Critical Infrastructures through SCADA Security Exploitation. *J. Secur. Eng*. **2009**, *2009*, 473–480.
15. Rezai, A.; Keshavarzi, P.; Moravej, Z. Secure SCADA communication by using a modified key management scheme. *ISA Trans*. **2013**, *52*, 517–524.
16. Kang, D.J.; Lee, J.J.; Kim, B.H.; Hur, D. Proposal strategies of key management for data encryption in SCADA network of electric power systems. *Int. J. Electr. Power Energy Syst.* **2011**, *33*, 1521–1526.
17. Lee, S.; Choi, D.; Park, C.; Kim, S. An efficient key management scheme for secure SCADA communication. Available online: http://www.waset.org/publications/15268 (accessed on 4 December 2014).
18. Hauser, C.H.; Bakken, D.E.; Bose, A. A failure to communicate: Next generation communication requirements, technologies, and architecture for the electric power grid. *IEEE Power Energy Mag.* **2005**, *3*, 47–55.
19. Pietre-cambacedes, L.; Sitbon, P. Cryptographic key management for SCADA systems-issues and perspectives. In Proceedings of the IEEE International Conference on Information Security and Assurance, Busan, Korea, 24–26 April 2008; pp. 156–161.
20. Xiao, L.; Yen, I.; Bastani, F. Scalable authentication and key management in SCADA. In Proceedings of the IEEE International Conference on Parallel and Distributed Systems, Shanghai, China, 8–10 December 2010; pp. 172–179.
21. Igure, V.M.; Laughter, S.A.; Williams, R.D. Security issues in SCADA networks. *Comput. Secur*. **2006**, *25*, 498–506.
22. Huitsing, P.; Chandia, R.; Papa, M.; Shenoi, S. Attack taxonomies for the MODBUS protocols. *Int. J. Crit. Infrastruct. Prot*. **2008**, *1*, 37–44.
23. Hong, S.; Lee, M. Challenges and Direction toward Secure Communication in the SCADA System. In Proceedings of the 2010 Eighth Annual Communication Networks and Services Research Conference (CNSR), Montreal, QC, Canada, 11–14 May 2010; doi:10.1109/CNSR.2010.52.
24. Pfleeger, C.; Pfleeger, S.L. *Security in Computing*; Prentice Hall: Upper Saddle River, NJ, USA, 2007.
25. Shahzad, A.; Musa, S.; Aborujilah, A.; Irfan, M. Secure Cryptography Testbed Implementation for SCADA Protocols Security. In Proceedings of 2013 International Conference on Advanced Computer Science Applications and Technologies (ACSAT), Kuching, Malaysia, 23–24 December 2013; pp. 315–320.
26. Ten, C.-W.; Liu, C.-C.; Govindarasu, M. Vulnerability Assessment of Cybersecurity for SCADA Systems Using Attack Trees. In Proceedings of the 2007 IEEE Power Engineering Society General Meeting, Tampa, FL, USA, 24–28 June 2007; doi:10.1109/PES.2007.385876.

27. Lee, D.; Kim, H.; Kim, K.; Yoo, P.D. Simulated Attack on DNP3 Protocol in SCADA System. In Proceedings of the 31th Symposium on Cryptography and Information Security, Kagoshima, Japan, 21–24 January 2014.
28. Fujisaki, E.; Okamoto, T. Secure integration of asymmetric and symmetric metric encryption schemes. *J. Cryptol.* **2013**, *26*, 81–101.
29. Graham, J.; Patel, S. *Security Considerations in SCADA Communication Protocols*; Technical Report TR-ISRL-04-01; Intelligent Systems Research Laboratory: Louisville, KY, USA, 2004
30. Shahzad, A.; Musa, S.; Irfan, M. N-Secure Cryptography Solution for SCADA Security Enhancement. *Trends Appl. Sci. Res.* **2014**, *9*, 381–395.
31. Neuman, B.C.; Ts'o, T. Kerberos: An authentication service for computer networks. *IEEE Commun. Mag*. **1994**, *32*, 33–38.
32. Kang, H.M.; Kim, A. Proposal for Key Policy of Symmetric Encryption Application to Cyber Security of KEPCO SCADA Network. In Proceedings of the Future Generation Communication and Networking (FGCN 2007), Jeju, Korea, 6–8 December 2007; Volume 2, pp. 609–613.
33. Nazri, M.; Alsharafi, A. Flooding Based DoS Attack Feature Selection Using Remove Correlated Attributes Algorithm. In Proceedings of 2013 International Conference on Advanced Computer Science Applications and Technologies (ACSAT), Kuching, Malaysia, 23–24 December 2013; pp. 93–96.
34. Khelil, A.; Germanus, D.; Suri, N. Protection of SCADA communication channels. In *Critical Infrastructure Protection*, Proceedings of the Critical Infrastructure Protection Lecture Notes in Computer Science; Springer Berlin Heidelberg: Berlin, Germany, 2012; Volume 7130, pp. 177–196.
35. Coates, G.M.; Hopkinson, K.M.; Graham, S.R.; Kurkowski, S.H. A trust system architecture for SCADA network security. *IEEE Trans. Power Deliv*. **2010**, *25*, 158–169.
36. Kim, H.J. Security and Vulnerability of SCADA Systems over IP-Based Wireless Sensor Networks. *Int. J. Distrib. Sens. Netw*. **2012**, *2012*, doi:10.1155/2012/268478.
37. Rush, W.F.; Kinast, J.A.; Shah, A.B. AGA 12 recommends how to protect SCADA communications from cyber attack. *Pipeline Gas J*. **2006**, *233*, 40. Available online: http://ipi.ir/standard/STANDS/AGA/AGA-12-SCADA.PDF (accessed on 4 December 2014).
38. Rezai, A.; Keshavarzi, P. High-performance modular exponentiation algorithm by using a new modified modular multiplication algorithm and common- multiplicand–multiplication method. In Proceedings of the IEEE World Congress on Internet Security, London, UK, 21–23 February 2011; pp. 192–197.
39. Ryu, D.H.; Kim, H.; Um, K. Reducing security vulnerabilities for critical infrastructure. *J. Loss Prev. Process Ind*. **2009**, *22*, 1020–1024.
40. Rezai, A.; Keshavarzi, P. High-performance implementation approach of elliptic curve cryptosystem for wireless network applications. In Proceedings of the 2011 IEEE International Conference on Consumer Electronic, Communication and Networks (CECNet), XianNing, China, 16–18 April 2011; pp. 1323–1327.
41. Riaz, R.; Naureen, A.; Akram, A.; Akbar, A.H.; Kim, K.H.; Ahmed, H.F. A unified security framework with three key management schemes for wireless sensor networks. *Comput. Commun*. **2008**, *31*, 4269–4280.

42. Dawson, R.; Boyd, C.; Dawson, E.; Nieto, J. SKMA, a key management architecture for SCADA systems. In Proceedings of the fourth Australasian information security workshop, Hobart, Australia, 16–19 January 2006; pp. 138–192.
43. Beaver, C.; Gallup, D.; Neumann, W.; Torgerson, M. *Key Management for SCADA*; Technical Report; Available online: http://www.sandia.gov/scada/documents/013252.pdf (accessed on 4 December 2014).
44. Choi, D.; Kim, H.; Won, D.; Kim, S. Advanced key management architecture for SCADA communications. *IEEE Trans. Power Deliv*. **2009**, *24*, 1154–1163.
45. Choi, D.; Lee, S.; Won, D.; Kim, S. Efficient secure group communications for SCADA. *IEEE Trans. Power Deliv*. **2010**, *25*, 714–722.
46. Choi, D.; Jeong, H.; Won, D.; Kim, S. Hybrid key management architecture for robust SCADA systems. *J. Inf. Sci. Eng.* **2013**, *29*, 281–298.
47. Rezai, A.; Keshavarzi, P.; Moravej, Z. A new key management scheme for SCADA network. In Proceedings of the 2nd International Symposium on Computing in Science and Engineering, Aydın, Turkey, 1–4 June 2011; pp. 373–378.
48. Almanei, S.; Alqattan, M.; Khamis, R.; Hussain, Y. *TCP/IP Protocol Possible Attacks*; Oregon State University: Corvallis, OR, USA, 2000; pp. 1–20.
49. Du, W.L. *Attack Lab: Attacks on TCP/IP Protocol*; Syracuse University: Syracuse, NY, USA, 2010; pp. 1–7.
50. Mander, T.; Nabhani, F.; Wang, L.; Cheung, R. Data Object Based Security for DNP3 Over TCP/IP for Increased Utility Commercial Aspects Security. In Proceedings of the 2007 IEEE Power Engineering Society General Meeting, Tampa, FL, USA, 24–28 June 2007; doi: 10.1109/PES.2007.386243.
51. Shahzad, A.; Musa, S.; Adulaziz, A.; Irfan, M. Industrial control systems (ICSs) vulnerabilities analysis and SCADA security enhancement using testbed encryption. In Proceedings of the 8th International Conference on Ubiquitous Information Management and Communication, Siem Reap, Cambodia, 9–11 January 2014.
52. Fovino, I.N.; Carcano, A.; Masera, M.; Trombetta, A. Design And Implementation of a Secure Modbus Protocol. In *Critical Infrastructure Protection III*, IFIP Advances in Information and Communication Technology; Springer Berlin Heidelberg: Berlin, Germany, 2009; Volume 311, pp. 83–96.
53. Byres, E.J.; Franz, M.; Mille, D. The use of attack trees in assessing vulnerabilities in SCADA systems. In Proceedings of the IEEE Conference on International Infrastructure Survivability Workshop (IISW ’04). Institute for Electrical and Electronics Engineers, Lisbon, Portugal, 4 December 2004. Available online: http://www.ida.liu.se/labs/rtslab/iisw04/camready/SCADA-Attack-Trees-Final.pdf (accessed on 12 April 2015).
54. kofahi. N.A. An Empirical Study to Compare the Performance of some Symmetric and Asymmetric Ciphers. *Int. J. Secur. Appl.* **2013**, *7*. Available online: http://www.sersc.org/journals/IJSIA/vol7_no5_2013/1.pdf (accessed on 12 April 2015).
55. Frankel, Y.; MacKenzie, P.D.; Yung, M. Robust efficient distributed RSA-key generation. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing STOC '98)*; ACM: New York, NY, USA, 1998; pp. 663–672.

56. Gao, J.; Liu, J.; Rajan, B.; Nori, R. SCADA communication and security issues. *Secur. Commun. Netw*. **2014**, *7*, 175–194.
57. Chae, H.; Shahzad, A.; Irfan, M.; Lee, H. Industrial Control Systems Vulnerabilities and Security Issues and Future Enhancements. *Adv. Sci. Technol. Lett.* **2015**, *95*, 144–148.
58. Majdalawieh, M.; Parisi-Presicce, F.; Wijesekera, D. DNPSec: Distributed Network Protocol Version 3 (DNP3) Security Framework. In *Advances in Computer, Information, and Systems Sciences, and Engineering*, Proceedings of IETA 2005, TeNe 2005, EIAE 2005; Springer: Houten, The Netherlands, 2006; pp. 227–234.
59. Jang, U.; Lim, H.; Kim, H. Privacy-Enhancing Security Protocol in LTE Initial Attack. *Symmetry* **2014**, *6*, 1011–1025.
60. Lu, T.B.; Guo, X.B.; Li, Y.; Peng, Y.; Zhang, X.Y.; Xie, F.; Gao, Y. Security for Industrial Control Systems Based on Wireless Sensor Networks. *Int. J. Distrib. Sens. Netw.* **2014**, *2014*, 1–17.
61. Rosa, T. Key-collisions in (EC) DSA: Attacking Non-repudiation. Extended version of the paper supporting a brief talk given at the CRYPTO 2002 Rump Session. Available online: https://eprint.iacr.org/2002/129.pdf (accessed on 12 April 2015).
62. Hernandez-Ardieta, J.L.; Gonzalez-Tablas, A.I.; de Fuentes, J.M.; Ramos, B. A taxonomy and survey of attacks on digital signatures. *Comput. Secur.* **2013**, *34*, 67–112.
63. Pajcin, B.R.; Ivanis, P.N. Analysis of Software Realized DSA Algorithm for Digital Signature. *Electronics* **2011**, *15*, Available online: http://electronics.etfbl.net/journal/Vol15No2/xPaper_12.pdf (accessed on 12 April 2015).
64. Chen, H.; Shen, X.; Wei, W. Digital Signature Algorithm Based on Hash Round Function and Self-Certified Public Key System. In Proceedings of the First International Workshop on Education Technology and Computer Science, Wuhan, China, 7–8 March 2009; Volume 2, pp. 618–624.
65. Kumar, H.; Singh, A. An Efficient Implementation of Digital Signature Algorithm with SRNN Public Key Cryptography. *IJRREST Int. J. Res. Rev. Eng. Sci. Technol*. **2012**, *1*, Available online: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.300.6509 (accessed on 12 April 2015).
66. Chen, H.; Shen, X.; Lv, Y. A New Digital Signature Algorithm Similar to ELGamal Type. *J. Softw*. **2010**, *5*, 320–327.
67. A. Shahzad, S.; Irfan, M. Deployment of New Dynamic Cryptography Buffer for SCADA Security Enhancement. *J. Appl. Sci*. **2014**, *14*, 2487–2497.
68. Test Harness, Triangle MicroWorks. Available online: www.trianglemicroworks.com (1 July 2015).
69. Saxena, A.; Pal, O.; Saiwan, S.; Saquib, Z. Token Based Key Management Scheme for SCADA Communication. *Int. J. Distrib. Parallel Syst.* **2011**, *2*, 69–86.
70. An Ettercap Primer. Available online: http://www.sans.org/reading-room/whitepapers/tools/ettercap-primer-1406 (accessed on 29 May 2015).
71. Ethereal. Available online: http://www.engr.siu.edu/~weng/ece553/wireshark-tutorial.pdf (accessed on 29 May 2015).
72. Aircrack. Available online: http://www.aircrack-ng.org/doku.php?id=cracking_wpa (accessed on 29 May 2015).
73. Air Snort. Available online: http://www.scribd.com/doc/50711790/airsnort-tutorial#scribd (accessed on 29 May 2015).

74. Packet Sniffer. Available online: https://www.mikrotik.com/testdocs/ros/2.9/tools/sniffer.pdf (accessed on 29 May 2015).
75. Dniffer. Available online: http://www.giac.org/paper/gsec/810/introduction-dsniff/101714 (accessed on 29 May 2015).
76. Airpwn. Available online: http://airpwn.sourceforge.net/Documentation.html (accessed on 29 May 2015).
77. File2air. Available online: http://www.willhackforsushi.com/?page_id=126 (accessed on 29 May 2015).
78. Son, S.; McKinley, K.S.; Shmatikov, V. Diglossia: Detecting Code Injection Attacks with Precision and Efficiency, **2013**. Available online: https://www.cs.utexas.edu/~shmat/shmat_ccs13.pdf (accessed on 29 May 2015).
79. Pietraszek, T.; Berghe, C.V. Defending against injection attacks through context-sensitive string evaluation. In *Proceedings of the 8th International Conference on Recent Advances in Intrusion Detection (RAID'05)*; Valdes, A., Zamboni, D., Eds.; Springer-Verlag: Berlin, Germany; Heidelberg, Germany, 2005; pp. 124–145.
80. Pinkas, B.; Sander, T. Securing Passwords Against Dictionary Attacks. Available online: http://www.pinkas.net/PAPERS/pwdweb.pdf (accessed on 29 May 2015).
81. Narayanan, A.; Shmatikov, V. Fast Dictionary Attacks on Passwords Using Time-Space Tradeoff. **2005**. Available online: https://www.cs.utexas.edu/~shmat/shmat_ccs05pwd.pdf (accessed on 29 May 2015).
82. Wireless Modbus TCP Communication. Available online: http://www.connectblue.com/press/articles/robust-wireless-modbus-tcp-communication/ (accessed on 29 May 2015).
83. White Paper. Process Control and Automation using Modbus Protocol. Available online: https://www.amplicon.com/docs/white-papers/MODBUS-in-Process-control.pdf (accessed on 29 May 2015).
84. R9120–1 ModHopper Wireless Modbus/Pulse Transceiver. Available online: http://www.chipkin.com/files/products/modhopper/R9120–1Cutsheet.pdf (accessed 29 May 2015).
85. ProSoft. Implementing Modbus TCP over Wireless. Available online: http://www.prosofttechnology.com/content/download/4598/32774/file/rlxihw_wireless+modbus+tcp_v31.pdf (accessed on 29 May 2015).
86. Application Note. Wireless Modbus Systems. Available online: http://www.mtl-inst.com/images/uploads/AN9033.pdf (accessed on 29 May 2015).
87. Wireless MeshScape Gateway—Wi-Modbus TCP. Available online: http://millennialnet.com/EnergyManagement/Products/Wi-Modbus-TCP.asp (accessed on 29 May 2015).
88. Shahzad, A.; Musa, S.; Irfan, M. Security Solution for SCADA Protocols Communication during Multicasting and Polling Scenario. *Trends Appl. Sci. Res.* **2014**, *9*, 396–405.
89. DNP Users Group. *DNP3 Application Layer Specification*, Version 2.00; DNP Organization: Washington, WA, USA, 2005; Volume 2.
90. DNP Users Group. *DNP3 Specification, Secure Authentication*; DNP Organization: Washington, WA, USA, 2010; Supplement to Volume 2.

91. Hieb, J.L.; Graham, J.H.; Patel, S.C. *Cyber Security Enhancements for SCADA and DCS Systems, ISRL-TR-07-02, Intelligent Systems Research Laboratory*; Technical Report TR-ISRL-07-02; University of Louisville: Louisville, KY, USA, 2007.
92. Hieb, J.; Graham, J.; Patel, S. Security Enhancements for Distributed Control Systems. In *Critical Infrastructure Protection*, IFIP International Federation for Information Processing; Springer US: New York, NY, USA, 2008; Volume 253, pp. 133–146.
93. Moral-Garcia, S.; Moral-Rubio, S.; Rosado, D.G.; Fernandez, E.B.; Fernandez-Medina, E. Enterprise security pattern: A new type of security pattern. *Secur. Commun. Netw*. **2014**, 1670–1690.
94. Irshad, A.; Sher, M.; Faisal, M.S. A secure authentication scheme for session initiation protocol by using ECC on the basis of the Tang and Liu scheme. *Secur. Commum. Netw*. **2014**, 1210–1218.
95. Lim, S.; Lee, E.; Park, C.-M. Equivalent public keys and a key substitution attack on the schemes from vector decomposition. *Secur. Commum. Netw*. **2014**, 1274–1282.
96. Drahansky, M.; Balitanas, M. Cipher for Internet-based Supervisory Control and Data Acquisition Architecture. *J. Secur. Eng*. **2011**. Available online: http://www.sersc.org/journals/JSE/vol8_no3_2011/1.pdf (accessed on 14 December 2014).
97. Shbib, R.; Zhou, S.; Alkadhimi, K. SCADA System Security, Complexity, and Security Proof. In *Pervasive Computing and the Networked World*; Springer Berlin Heidelberg: Berlin, Germany, 2013; pp. 405–410.
98. Johnson, R.E. Survey of SCADA security challenges and potential attack vectors. In Proceedings of 2010 International Conference for Internet Technology and Secured Transactions (ICITST), London, UK, 8–11 November 2010; pp.1–5.
99. Heo, S.; Lee, S.; Doo, S.; Yoon, H. Design of a Secure System Considering Quality of Service. *Symmetry* **2014**, *6*, 938–953.
100. Moon, D.; Im, H.; Lee, J.; Park, J. MLDS: Multi-Layer Defense System for Preventing Advanced Persistent Threats. *Symmetry* **2014**, *6*, 997–1010.
101. Jung, M. A Study on Electronic-Money Technology Using Near Field Communication. *Symmetry* **2015**, *7*, 1–14.
102. Nam, J.; Choo, K.; Han, S.; Paik, J.; Won, D. Two-Round Password-Only Authenticated Key Exchange in the Three-Party Setting. *Symmetry* **2015**, *7*, 105–124.
103. He, D.; Chen, J.; Chen, Y. A secure mutual authentication scheme for session initiation protocol using elliptic curve cryptography. *Secur. Commun. Netw*. **2012**, *5*, 1423–1429.
104. Chandia, R.; Gonzalez, J.; Kilpatrick, T.; Papa, M.; Shenoi, S. Security Strategies for SCADA Networks. In *Critical Infrastructure Protection*, *IFIP International Federation for Information Processing*; Springer US: New York, NY, USA, 2008; Volume 253, pp. 117–131.
105. Rong, C.; Nguyen, S.T.; Jaatun, M.G. Beyond lightning: A survey on security challenges in cloud computing. *Comput. Electr. Eng*. **2013**, *39*, 47–54.
106. Chen, Y.; Dong, Q. RCCA security for KEM+DEM style hybrid encryptions and a general hybrid paradigm from RCCA-secure KEMs to CCA-secure encryptions. *Secur. Commun. Netw*. **2014**, *7*, 1219–1231.

107. Liyanage, M.; Gurtov, A. Securing virtual private LAN service by efficient key management. *Secur. Commun. Netw*. **2014**, *7*, 1–13.
108. Li, J.; Lin, Y.; Wang, G.; Li, R.; Yin, B. Privacy and integrity preserving skyline queries in tiered sensor networks. *Secur. Commun. Netw*. **2014**, *7*, 1177–1188.
109. Raza, S.; Duquennoy, S.; Höglund, J.; Roedig, U.; Voigt, T. Secure communication for the Internet of Things—A comparison of link-layer security and IPsec for 6LoWPAN. *Secur. Commun. Netw.* **2014**, *7*, 2654–2668.