Symmetry **2015**, *7*, 1945-1980; doi:10.3390/sym7041945

OPEN ACCESS

symmetry

ISSN 2073-8994

www.mdpi.com/journal/symmetry

Article

New Security Development and Trends to Secure the SCADA Sensors Automated Transmission during Critical Sessions

Aamir Shahzad [1], Malrey Lee [1,*], Hyung Doo Kim [2,*], Seon-mi Woo [3] and Naixue Xiong [4]

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

[2] Department of Fire Service Administration, Wonkwang University, Iksan 570-749, Korea

[3] JINI Co. Ltd., B-102, Technobill, 109 banryong-load, Deokjin Gu, JeonJu si, Jeollabuk-do 561-756, Korea, E-Mail: smwoo@jbnu.ac.kr

[4] School of Computer Science, Colorado Technical University, CO 80907, USA; E-Mail: xiongnaixue@gmail.com

***** Authors to whom correspondence should be addressed; E-Mails: mrlee@chonbuk.ac.kr (M.L.) khd4064@naver.com (H.D.K.); Tel.: +82-10-3611-8004 (M.L.); +82-63-270-3993 (H.D.K.).

Academic Editor: Sergei Odintsov

Received: 30 May 2015 / Accepted: 13 October 2015 / Published: 23 October 2015

Abstract: Modern technology enhancements have been used worldwide to fulfill the requirements of the industrial sector, especially in supervisory control and data acquisition (SCADA) systems as a part of industrial control systems (ICS). SCADA systems have gained popularity in industrial automations due to technology enhancements and connectivity with modern computer networks and/or protocols. The procurement of new technologies has made SCADA systems important and helpful to processing in oil lines, water treatment plants, and electricity generation and control stations. On the other hand, these systems have vulnerabilities like other traditional computer networks (or systems), especially when interconnected with open platforms. Many international organizations and researchers have proposed and deployed solutions for SCADA security enhancement, but most of these have been based on node-to-node security, without emphasizing critical sessions that are linked directly with industrial processing and automation. This study concerns SCADA security measures related to critical processing with specified sessions of automated polling, analyzing cryptography mechanisms and deploying the appropriate explicit inclusive security solution in a distributed network protocol version 3 (DNP3) stack, as part of a SCADA system. The bytes flow through the DNP3 stack with security

computational bytes within specified critical intervals defined for polling. We took critical processing knowledge into account when designing a SCADA/DNP3 testbed and deploying a cryptography solution that did not affect communications.

Keywords: supervisory control and data acquisition; distributed network protocol; dynamic cryptography buffer; integrity and event polling

1. Introduction

Supervisory control and data acquisition (SCADA) systems have been playing crucial roles in industrial automation and control. SCADA systems contribute to several processes including industrial production, refining, filtration, manufacturing and electric or power generation in industries such as automotive; heating, ventilating, and air conditioning (HVAC) and heat recovery (HR) ventilation/energy recovery ventilators (ERVs); oil and gas; water pumping, treatments, and distribution; aircraft and trains; and electricity generation, transmission and distribution [1–4]. The SCADA systems employ several protocols such as Modbus, IEC protocol series, Fieldbus, Profibus, Omnibus, DNP3, and Conitel, and each of these protocols has been designed for a specific industry, although a few of them are employed in multiple industries or industrial processes [1,2,4]. Figure 1 graphically depicts the general architecture of a SCADA system with its major components.

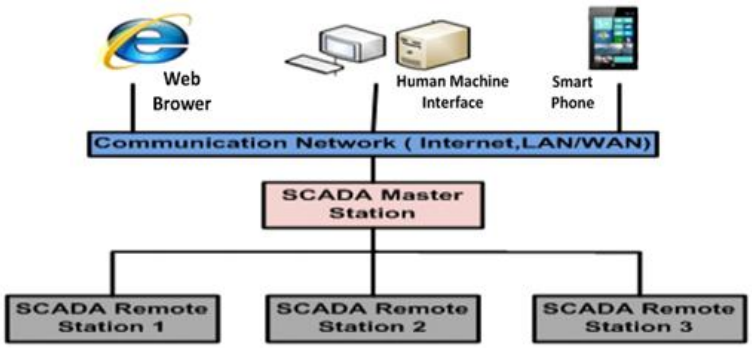


Figure 1. Supervisory control and data acquisition (SCADA) system architecture.

This study focuses on DNP3 and its open interconnectivity, and the major vulnerabilities that have been arising in SCADA/DNP3 communication. The DNP3 is an important SCADA communication protocol that is employed and designed for electric and water industries. In the SCADA system, DNP3 performs several operations followed by functions (codes), such as transfer functions, control functions, freeze functions and time synchronization functions, and file transferring functions [5]. Now-a-days, DNP3 is employed to interconnect field devices or remote sensors, located in various geographical locations and controlled from central controller(s) [6–8]. With the acquisition of modern technologies and connectivity over the Internet, SCADA/DNP3 communication has been vulnerable to cyber security threats which can be tremendously harmful to industrial systems [7–9].

SCADA/DNP3 uses the concept of polling, which is usually classified into two polling sections: integrity polling and event polling. In integrity polling, configured field devices (or remote stations)

respond to all static points, which have been observed most recently; while event polling is designated for significant changes occurring within the system or during the last integrity poll and report by exception [2,5]. The time factor has been considered as the most important concern during SCADA critical processing or polling. During polling, the session has been specified between network nodes, in which bytes are transmitted. Each node in the SCADA network should send/receive the bytes at a specified time interval. Therefore, a specified time is crucial for each node during SCADA transmission [6,10]. Several security designs for that type of transmission are inappropriate due to time limitations [6,10–12].

Security is an important aspect for all types of networks or systems in the arena of computer science and information technology. Several security mechanisms, including security pattern approaches, TLS/SSL, Internet Protocol Security (IPSec), Secure Shell (SSH), hardware/software firewalls, demilitarized zones (DMZs), authentication protocols, and cryptography solutions, have been designed and deployed to protect industrial communication against vulnerabilities and potential attacks [6,7,13–15]. Among these security approaches, cryptography based mechanisms are suggested as the best security approaches, due to their independence and implementation evaluation [8,16–19]. SCADA security via cryptography mechanisms has been considered as a vital approach during transmission [19–22]. On the other hand, cryptography based approaches are deemed to be complex and heavy computational approaches during security design and development—especially, in industrial processing [8,10,19,22–25]. The proposed study has analyzed the cryptography approaches, and then, employed the inclusive security solution in DNP3 which significantly increased the security of the SCADA system. The main contribution of this study is twofold:

(i) The SCADA/DNP3 polling scenario called automated polling is addressed and, according to communication requirements, security is deployed to secure the sensitive information. The sensitive information would be secured before travelling to non-proprietary protocols over the Internet.
(ii) To achieve security, the DNP3 stack is designed with an open source library, the original stack bytes are controlled and manipulated by new 56-byte dynamic development called a dynamic cryptography buffer (DCB), without changing the original protocol design.

The rest of the research paper is organized as follows. The DNP3 and its layers are explained in Section 2. We discuss the problem statement and research objectives in Section 3. In Section 4, the detailed DNP3 model is designed with security development and formal proofs are employed for validation purposes. DCB is deployed in Section 5. The testbed is setup in Section 6 and protocol bytes are flowed in Section 7. Measurements are observed and discussed in Section 8. Section 9 conducts a detailed survey of existing studies and Section 10 concludes our proposed research and future trends.

2. DNP3

In the SCADA system, DNP3 is the most demanding telecommunications standard based protocol, which provides communication facilities between main or supervisory stations, remote terminal units, and other intelligent equipment, in water and electric companies [1]. The DNP3 protocol was designed as a proprietary protocol and provides interoperability between various types of equipment. Seventy percent of DNP3 has been employed in North America, and the remaining 30% in the rest of the world,

notably South America, Africa, Australia, Asia and Europe [1,2]. Due to growing demands, DNP3 is now also connected with the Internet, and sends/receives the information that may be available geographically at remote sites through connected field devices. In addition, the non-proprietary protocols, such as TCP/IP (Transmission Control Protocol/Internet Protocol) and User Datagram Protocol (UDP), are used. These protocols provide interconnectivity with DNP3 over the Internet. Therefore, this protocol is also defined as an open or non-proprietary protocol in SCADA systems [1–3,6–11].

The enhanced performance architecture (EPA) model is a three-layer model, defined by the International Electrotechnical Commission (IEC) and the DNP3 design is also based on this EPA model. The EPA model is based on the Open Systems Interconnection (OSI) model that is a seven-layer model, and contains an application layer, data link layer and physical layer. DNP3 also uses these three layers, plus an additional layer called the pseudo-transport layer which performance the limited functions of transport layer and network layer of OSI model [1,2].

In DNP3, the application layer is a top layer that is designed to take the information from the upper layer (or user application layer). The user layer could form a human machine interface (HMI) or other SCADA/DNP3 supported software. Information passes from the user application layer to the application layer of DNP3. Here, variable sized data is managed in fixed sized blocks (or manageable sized data) and the addition of the application header forms fragments. The number of blocks or application service data units (ASDUs) is not limited in a fragment, but the size of fragment or application protocol data unit (APDU) is limited to 2048 bytes, which is also specified by original DNP3 documentation. The send or receive message is also specified at an application layer by means of the application protocol control information (APCI) or header fields. Sending APCI contains two bytes of information. This information includes application control (AC) and function code (FC) while the response APCI adds an additional field of two bytes called internal indication (IIN). The DNP3 protocol message contains various function codes that would be performed by the sub-controller, and would reply to the main controller [5]. Table 1 shows the number of application layer function codes that are performed in the SCADA/DNP3 system.

Table 1. Application layer function codes [3].

Function Type	Function Code	Function Perform
Request Function Codes		
Transfer Function	0	Confirm
	1	Read
	2	Write
Control Function	3–6	–
Freeze Function	7–12	–
Application Control Function	13–18	–
Configuration Function	19–22	–
Time Synchronization	23	–
Reserved	24–128	–
Response Function Codes		
Response Function	0	Confirm
	129	Read
	130	Write

In the pseudo-transport layer, APDU bytes are assembled as a transport service data unit (TSDU) and TSDU is disassembled into fixed sized data blocks, except in the case of the last block of 56 bytes, when the maximum bytes are received from the application layer or 2048 bytes of APDU. The transport layer adds one byte of header with each data block, and transport protocol data unit (TPDU) or a segment is formed. Each TPDU size is limited to a maximum of 250 bytes, which would be further employed in the link layer frame. The data link layer assembles each upcoming TPDU as a link service data unit (LSDU), and adds a header field of 10 bytes, which is also called link protocol control information (LPCI). A link protocol data unit (LPDU) (or frame) is formed by adding the LPCI with an LSDU block. In the data link layer, source and destination addresses are defined and a 32 bytes cyclic redundancy checker (CRC) code is employed to detect the transmission errors [1–3]. More detail of DNP3 and its related layer fields is illustrated in Figure 2.

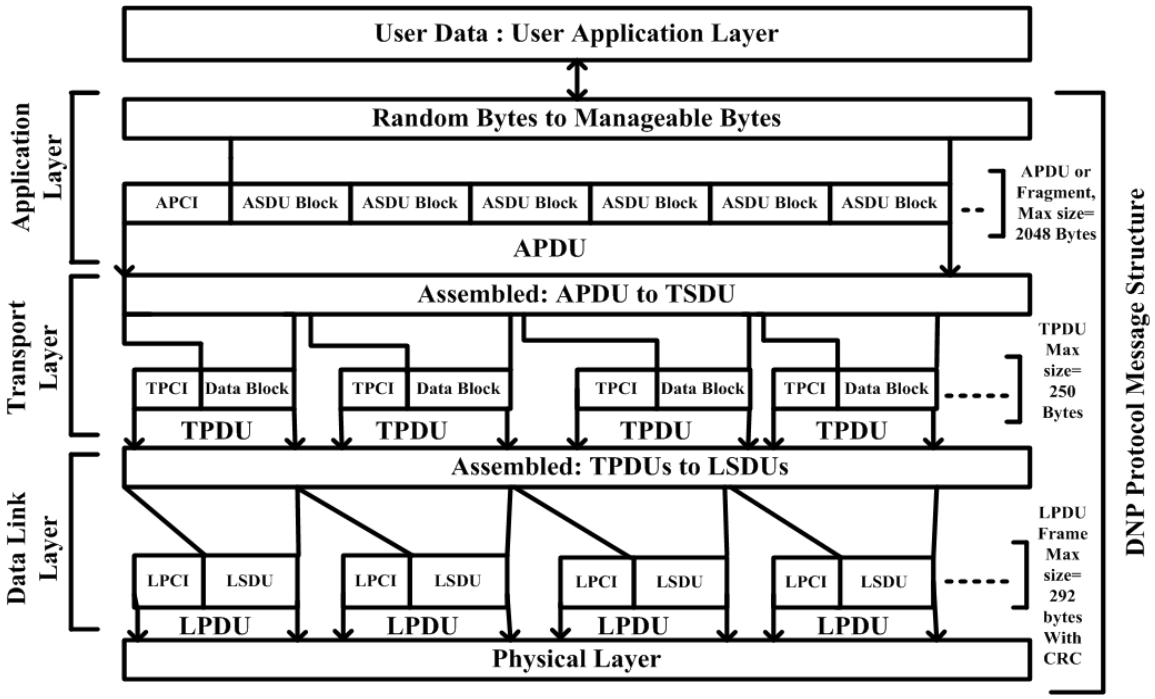


Figure 2. DNP3 protocol stack and related fields [3].

The link layer frames are then transmitted to the physical layer. As defined, DNP3 is a non-proprietary protocol. Therefore, the frames are directly encapsulated into TCP/IP protocols which provide a pathway to DNP3 frames to travel over the Internet. At the receiver side, frames are reassembled into TPDU (blocks), by a stripped of link header or LPCI. After this, the transport header or TPCI is also removed from each TPDU or segment, and then, the TSDU blocks are reformed. Each TSDU block will assemble as APDU or an application layer fragment. The header bytes are verified and removed, and the ASDU blocks are reformed and these blocks would be used at the user application layer [3]. Figure 3 shows the SCADA/DNP3 stack and its connectivity with TCP/IP protocols.

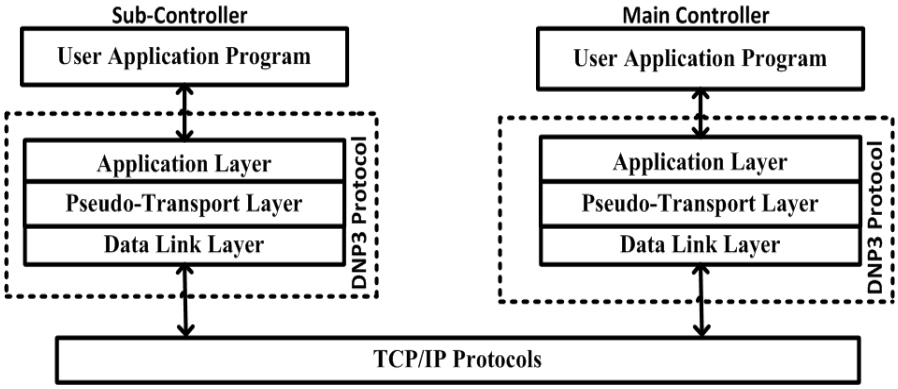


Figure 3. SCADA/DNP3 connectivity with TCP/IP protocols.

3. Problem Statement

3.1. Background Study

In the initial structure of DNP3, there is no security mechanism that can protect sensitive information from security threats [5,7,21,23]. Due to the evolution of technology, the advanced version of DNP3 provides interconnectivity over the Internet; the information travels through non-proprietary protocols, which reside below DNP3 [5,19]. Due to open connectivity, DNP3 has been vulnerable to Internet attacks; most DNP3 devices are configured, and communicate without any proper authentication mechanism or have little protection in the SCADA network against vulnerabilities [23–30]. Cryptography based security mechanisms [31] have been proposed for DNP3 by DNP3 users group, in which symmetric and asymmetric methods are defined and a detailed description of challenge-response technique is made to examine the security goals (or parameters), such as authentication and integrity, and to protect the transmission against attacks, such as replay, spoofing, and modification attacks [8,32,33], at the application layer. However, many limitations are accounted for in DNP3 security design and development, and most of the work is in initial phases or still in the development phases.

DNP3 provides three main communication facilities (or modes) to connect the field devices in SCADA networks including unicasting (or master or outstation mode), broadcasting mode and unsolicited mode. A unicasting mode is also designated for a peer-to-peer mode, in which the main controller requests information, and the sub-controller will reply. In broadcasting, the main controller broadcasts the information to all connected sub-controllers in the SCADA/DNP3 network(s). When necessary, sub-controllers are authorized to send unsolicited responses to the main controller. The message may be an alternative message to the main controller. In communication modes, DNP3 used TCP/IP protocols to communicate over the Internet, as well as the available security protocols (*i.e.*, TLS/SSL and IPSec) to protect against unauthorized threats. There is no proper authentication mechanism in DNP3. Therefore, DNP3 frames are encapsulated in other security protocols including TLS/SSL and IPSec. A survey has been conducted on SCADA/DNP3 vulnerabilities [24–27,31], and mechanisms [32–34], such as anomaly detection and attacks detection, are used to detect the attacks, such as flooding attacks, DoSattack, spoofing, data modification, data reply, and man-in-the-middle attack, in SCADA/DNP3 communication [12,33,35,36]. A number of attacks is investigated [20,33,35–40] by employing attack scenarios in the SCADA/DNP3 system, and the existence and potential influence of attacks are also

measured. Another study [27] incorporated 28 potential attacks in the SCADA/DNP3 system due to protocol deficiencies.

3.2. Study Motivation

The existing work [6,7,15–21] of SCADA security has employed the end-to-end developments to secure the communication of SCADA systems. A number of limitations and dependencies are found during end-to-end developments [8,19,27]. More specifically, the DNP3 frames are encapsulated in other lower layer protocols, such as SSH, IPSec, and SSL/TLS [23,41], which protect the sensitive information of the SCADA/DNP3 system against Internet vulnerabilities and attacks [20,39,42–47]. In conclusion, DNP3 relies on other open protocols, such as TCP/IP and UDP, in terms of transmission over the Internet and on protocols (such as SSH, IPSec, and SSL/TLS) for security purposes, but these open protocols (such as TCP/IP and UDP) have several vulnerabilities, and the protocols (such as SSH, IPSec, and SSL/TLS) also have limitations [8,36], because they depend on other security protocols, such as cryptography protocols [19,23].

Research [16–23] has been conducted on the vulnerable aspects of SCADA systems. Specifically, cryptography mechanisms have been developed to enhance the security of these critical systems, and most of these developments have been successful. Few cryptography mechanisms [12,23,24,48] are proposed for SCADA/DNP3 systems in case of one-to-one communication, and are inappropriated while employing public key cryptography in broadcasting communication. Public key cryptography based mechanisms required much time during key generation, distribution, and algorithm computation [16,17,22,48–50]. On the other side, symmetric cryptography solutions are unable to protect SCADA communication against non-repudiation attacks [8,16,23].

From the above SCADA/DNP3 security analysis, we can conclude that DNP3 lacks security, even when employing and depending on other protocols, which we examine in order to better understand the current security of DNP3. Therefore, this study proposes work trends to secure the SCADA/DNP3 system, and highlights and addresses the potential security measurements such as authentication, integrity and confidentiality during polling scenarios. To achieve the desired goals, the proposed study fulfills the following main objectives:

(i) The SCADA/DNP3 stack has been designed using an open source library with explicit codes in C#, and security is implemented within the stack, before it communicates with open protocols.

(ii) A new automated polling scenario has been designed that combines the balanced and unbalanced systems of DNP3, according to the requirements of the organization in a water pumping system.

(iii) In the original DNP3 stack, bytes are constructed followed by layer(s) specifications, and security is deployed, and then, bytes are placed in a dynamic cryptography buffer (DCB) for further development. In security development, well-known security algorithms are selected from the arena of cryptography, such as advanced encryption standard (AES) and SHA-2, which significantly enhance SCADA/DNP3 security without interrupting the communication (or polling) specifications.

(iv) A simulation environment is designed for the SCADA system by employing DNP3; bytes are constructed with security development, and are transmitted a number of times between controllers in the SCADA/DNP3 network.
(v) Formal proofs are used which validate the proposed research, including the validation of the bytes construction processes within the stack, validation of security implementation and validation of DCB design and development.
(vi) Well-known tools (or attacking tools) are used that interrupt the normal communication of the SCADA/DNP3 system that we can use to measure and evaluate performance.
(vii) An evaluation process is performed based on three measurements phases:

(a) The DNP3 bytes are constructed without security concerns, and transmitted to an open network. Attacking tools are used to interrupt the normal communication, and security performance is measured.
(b) In the second phase, proposed security is implementing via DCB, and performance results are computed.
(c) In the last phase, cryptography algorithms such as AES and SHA-2 are deployed and tested at each end of SCADA/DNP3 system and performances are computed that would be helpful during the comparison process.

The above three measurements are helpful during the comparison process, as well as at the time of evaluation. The measured results of the second phase are further compared with existing developments (results) that would show the difference between them.

The scope of this study is limited to SCADA/DNP3 security designed and deployed during a critical scenario called automated polling. The most prominent cryptography algorithms are considered to secure the communication against attacks, including confidentiality, integrity and authentication. The proposed security development is selected on the basis of polling specifications. The security keys are locally stored and are distributed (or exchanged) statically among the participated nodes in the testbed, which are considered as the limitations of this study. Therefore, in future, certificate authority (CA) is required for digital certification in the case of public key cryptography, and a key distribution center (KDC) (*i.e.*, Kerberos) is required in the case of symmetric key cryptography where keys are exchanged among participated nodes using secure channel(s).

4. Model Design and Development

The polling is an important concept used in SCADA/DNP3 systems: balanced systems and unbalanced systems [3,5]. In this study, we review the existing polling scenarios and then we change the logical polling (or polling) to automated polling to address the security of a water pumping system. A simulation environment is designed for a SCADA water pumping system, in which several sub-controllers are used to receive the information from a pumping system through sensors, such as pressure sensors, heating sensors, cooling sensors, level sensors, and other equipment.

In automated polling, explicit parameters are set with time constraints at specific intervals. In other words, the SCADA/DNP3 main controller polls once each time and sub-controllers respond according to parameters at specified intervals. After completing an interval, the SCADA/DNP3 main controller

will again poll in the event of new command execution, otherwise, only the interval will be changed. During automated polling, security is an important issue. Therefore, a reliable security mechanism is required that significantly prevents the system against potential attacks.

For example, the main controller polls the sub-controller(s) to send the status of the water level, and the corresponding heating measured points at an interval of 7 min. This means that the parameters are as follows: an integrity poll response is issued every 45 s, a response to a normal event is sent every 20 s, and a reply to an abnormal event is sent depending on the situation.

In this study, the event polling is divided into two parts: normal event polling and abnormal event polling. In normal event polling, a message such as the delivery time of the first integrity poll and then the time difference between integrity polls and the sequence number and transmission status are employed in the functional field called non-critical. At the other side, abnormal event polling acts as an unsolicited response event, in which the response would be sent any time. The response events are: exception report, alter message, change measured in the last integrity poll response, and the status of the unknown entity that interrupts in the polling. These events are also called critical response events (or critical response events bytes).

The DNP3 model and its security development are thoroughly explained in the section below. Formal definitions and postulates are employed, which validate the proposed design and development. After this, DCB is employed to track and control model bytes, and other special operations of security development.

4.1. DNP3 Model Design and Security Development

In security development, constructed protocol bytes are treated as user bytes. Two cryptography algorithms are employed; the Advanced Encryption Standard (AES) and SHA-2 are deployed at the data link layer and an SHA-2 hashing algorithm is deployed at the pseudo-transport layer and application layer of DNP3. The cryptography algorithms and related design are selected based on communication requirements. As security development is dynamic, we can change the design according to the end user and SCADA system demands.

In Figure 4, there are logical bytes, which are constructed at each layer in the DNP3 stack. Each cell has one byte, represented in a hexadecimal format. At the application layer, the bytes in black are application service data unit (ASDU) constructed bytes, and an empty cell shows application protocol control information (APCI) such that, red bytes: “C3” is the application code, “01” is the function code, and the remaining bytes, such as “00” and “00”, represent the Internet Indication (IIN), in case of a response message. The bytes such as “1A” and “EE” are security development bytes; employing the SHA-2 hashing function provides security against integrity attacks.

In the pseudo-transport layer, the application layer bytes are assembled into transport service data unit (TSDU) and a one-byte header field is added with TSDU bytes, which made the full transport protocol data unit (TPDU). Each TPDU contains a maximum of 250 bytes. Therefore, the upper layer bytes (or application layer bytes) easily fit into one TPDU, as visualized in Figure 4. The black bytes are TSDU bytes. The empty cell shows the transport protocol control information (TPCI), and is represented by red byte “OB”. The SHA-2 hash digest of TPDU bytes is calculated and represented by red bytes, such as “2A” and “EE”, which provide byte verification and protect the contents from unknown entities.

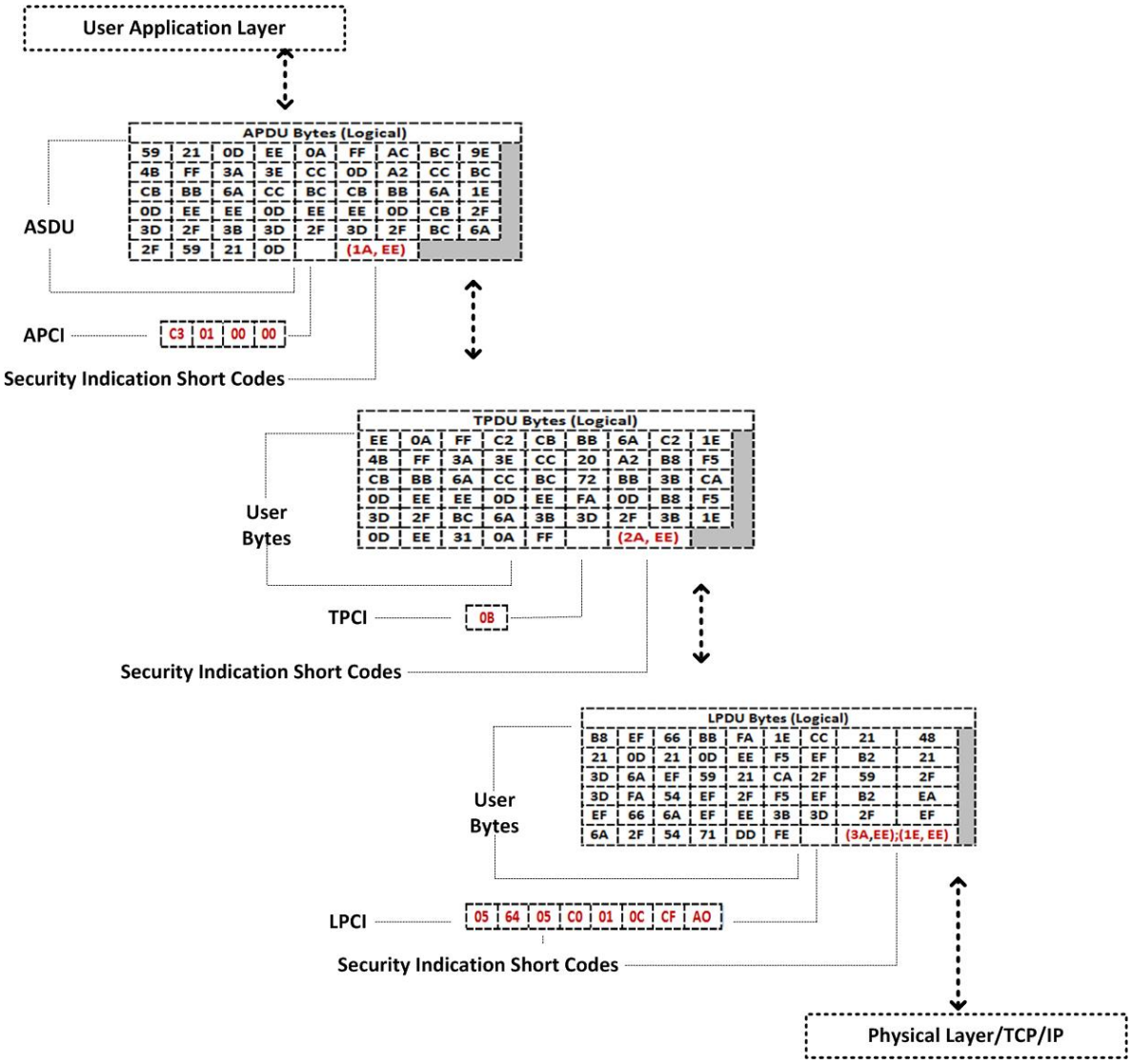


Figure 4. Logical bytes in the DNP3 stack.

In the data link layer, the upper layer bytes are assembled as link service data unit (LSDU) and a 10 bytes header is added, which forms the link protocol data unit (LPDU). The maximum of LPDU is 260 bytes, plus 32 CRC bytes. In this study, 32 CRC bytes are treated as optional bytes, or could be required as additional bytes for DCB. In Figure 4, in the LPDU buffer, the bytes in black are link service data unit (LSDU) constructed bytes, and the empty cell shows the link protocol control information (LPCI). Details of the red bytes are as follows: "05"and "64" are a start of link header, "05" is length, "C0" is control byte, "01" is the destination address, "0C" is the source address, and the remaining bytes such as "CF" and "A0" represent the optional CRC bytes, in the event of sending/responding to a message. The remaining bytes, such as "3A" and "EE", represent the security development bytes by employing an SHA-2 hashing function while the bytes, such as "1E" and "EE", are computed by an AES algorithm, which provides security against integrity, authentication, and confidentiality attacks.

The bytes are encrypted by employing an AES algorithm rather than by appending the secret code (or key) with protocol constructed bytes. The bytes' encryption is a good approach, and this cannot affect the other bytes in the protocol stack [8]. In DNP3, upper layer bytes are counted as user bytes in the lower layer [3].

In each layer, security bytes are computed and represented by codes of two bytes, including "1AEE" in APDU, "2AEE" in TPDU and "3AEE" and "1EEE" in LPDU. In other words, these are short codes that designate the security algorithms employed from cryptography, and at the receiver side, security development will be required. The proposed study also scrutinizes the deployment of a symmetric algorithm at the application layer. This is a good approach to secure the information, in case the attacker aims to target the application layer (buffer). However, due to communication specifications (or polling specifications) such as time limitations, the AES algorithm is only deployed at the data link layer. However, the attack scenarios are employed to launch the attacks by considering the whole protocol stack rather than taking into account the specific layer of DNP3. In the future, an attacks scenario will be designed, and launched at each layer of DNP3; this would be a good alternative approach to test the vulnerabilities of the desired layer(s). Figure 5 shows the security development in the DNP3 protocol stack.

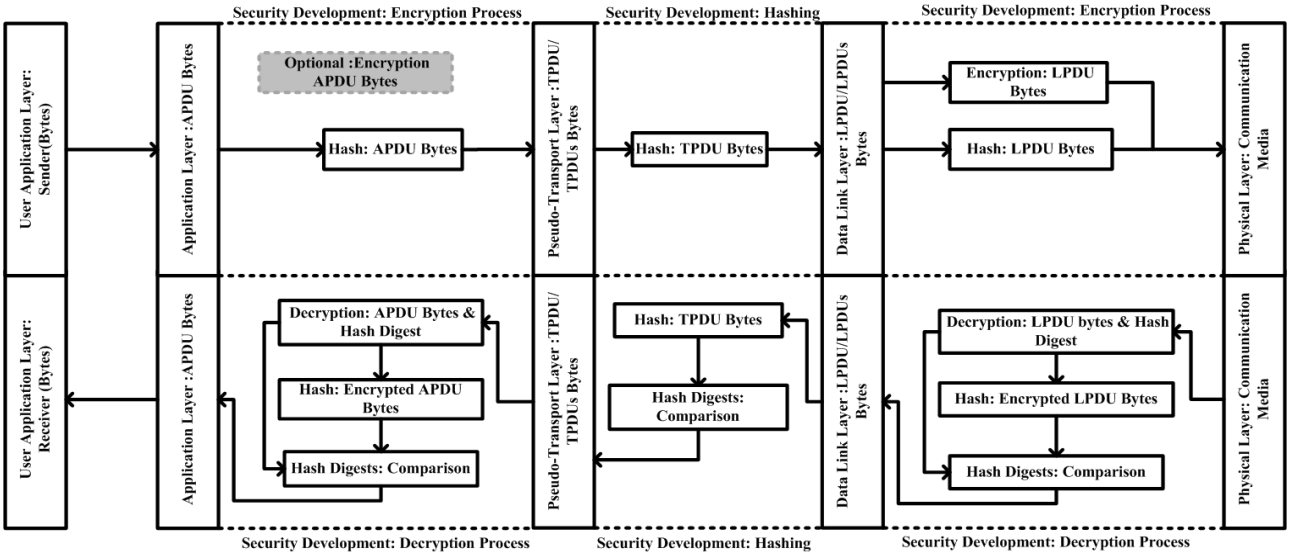


Figure 5. Security development within the DNP3 stack.

In conclusion, security is deployed within the stack so the proposed security development acts as an additional security layer(s), which performs a security test before assembling, reassembling and disassembling bytes in the protocol stack. In other words, three additional security layers are embedded inside the stack, which protects the sensitive information from unauthorized users. This would also be helpful if the attacker individually attacks DNP3 layers, such as the application layer, pseudo-transport layer and data link layer, to retrieve the sensitive information; for example, if the attacker has planned to retrieve the sensitive information of the application layer, without concern for the other layers. The security layer (at the application layer) protects the bytes against attackers. This situation would also arise in the other layers.

4.2. Model Definitions

DNP3 is designed without any security concerns. Therefore, a new DNP3 stack model is designed with security development. The security is achieved; if there is a protocol, constructed bytes flow through the stack (*i.e.*, DNP3 stack) to non-proprietary protocols (*i.e.*, TCP/IP).

Definition 1 (Manageable Bytes). A set of bytes "X" or user payload is observed from the user application layer using function f_{μ}; such that $f_{\mu_0} \in f_{\mu}$ is an independent function for each transmission by employing distinguishing identifiers.

A function f_{μ_0} gets the bytes from the user layer or user application layer and grants them to the application layer of DNP3, x ∈ X at runtime. The total size of X is limited according to the protocol design and specifications. The size is limited up to 1992 bytes, and the remaining bytes are employed for DCB.

$\forall X.\, \mu_{(E[k])}$, the μ takes the numbers of user specified bytes X and processes these bytes as protocol useable bytes, in the application layer. The detailed description of keys (k) has been added later, by employing $E[k]$. The function f_{μ_0} is executed for both the sender and receiver payload, while identifiers are used to distinguish them.

Definition 2 (Header Bytes). Two explicit functions: $f(\mu_1)$ and $f(\mu_2)$ are used to specify the protocol descriptors, which would be further employed in lower layers. Such that: $f(\mu_1, \mu_2) \in f_{\mu}$.

$$f(\mu_1, \mu_2) \Leftrightarrow \forall X. \sum_{k=0}^{n} \left(f_{(\mu_1,\mu_2)}: (\mathrm{X}_h), (\mathrm{X}_d)\right)$$

$$\Rightarrow \left(f_{\mu_{(1.1,1.2)}}: X_{ac}^{h}, X_{fc}^{h}\right) \in f_{(\mu_1)}(\mathrm{X}_h) \wedge (f_{\mu_{(1.1)}}, f_{\mu_{(1.2)}}) \in f(\mu_1), \text{ In-case of sending payload.}$$

$$\vee \left(f_{\mu_{(1.1,1.2,1.3)}}: X_{ac}^{h}, X_{fc}^{h}, X_{iin}^{h}\right) \in f_{(\mu_1)}(\mathrm{X}_h) \wedge (f_{\mu_{(1.1)}}, f_{\mu_{(1.2)}} f_{\mu_{(1.3)}}) \in f(\mu_1), \text{ In-case of response payload.}$$

$$\wedge \left(f_{\mu_{(2.1,2.2,2.3)}}: X_{oh}^{d}, X_{do}^{d}\right) \in f_{(\mu_2)}(\mathrm{X}_d) \wedge (f_{\mu_{(2.1)}}, f_{\mu_{(2.2)}} f_{\mu_{(2.3)}}) \in f(\mu_2), \text{ data blocks within the application layer stack.}$$

$$\Rightarrow X \leq \mathrm{X}_d \geq \mathrm{X}_h \wedge (\mathrm{h}, \mathrm{d}) \leq \text{Limit}$$

The application protocol data unit (APDU) is generated by combining two functions. Such that: $f: f_{(\mu_1)}(\mathrm{X}_h), f_{(\mu_2)}(\mathrm{X}_d)$. The header ($h$) and data ($d$) bytes are processed according to the application layer stack specifications and limitations, while the APDU size has decreased due to security information storage.

$$\exists\, f: f_{(\mu_1)}(\mathrm{Y}_h), f_{(\mu_2)}(\mathrm{Y}_d) \wedge \exists\, f: f_{(\mu_1)}(\mathrm{Z}_h), f_{(\mu_2)}(\mathrm{Z}_d) \wedge f(\mu_1, \mu_2) \in f_{\mu}.$$

The values, $(\mathrm{Y}_h, \mathrm{Y}_d)$ and $(\mathrm{Z}_h, \mathrm{Z}_d)$ are specified for the transport protocol data unit (TPDU) and link protocol data unit (LPDU).

Definition 3 (Flow of Bytes in Stack). A function $f(\mu_3)$ has been employed to manage the flow of bytes within the stack during protocol message construction and the security development process. Such that: $f(\mu_3) \in f_{\mu}$.

$$f(\mu_3) \Leftrightarrow \forall X. \sum_{\text{value}}^{\text{limit}} X_{b^{dy}} \Leftrightarrow b \leq \text{value} \leq \text{Limit} \wedge \text{value} \leq dy \leq \text{Limit}$$

$$\Rightarrow \forall X. \sum_{k=0}^{n} \left(f_{(\mu_1,\mu_2)}: (\mathrm{X}_h), (\mathrm{X}_d)\right) \,||\, \forall X. = \sum_{\text{value}}^{\text{limit}} X_{b^{dy}} \leq \text{ Limit, Single APDU}$$

$$\Rightarrow \forall X. \sum_{k=0}^{n} \left(f_{(\mu_1,\mu_2)}: (\mathrm{X}_h), (\mathrm{X}_d)\right) \,||\, \forall X. = \sum_{\text{value}}^{\text{limit}} X_{b^{dy}} \geq \text{ Limit, Multiple APDU}$$

$$\Rightarrow \forall X. \sum_{k=0}^{n} \left(f_{(\mu_1,\mu_2)}: (\mathrm{X}_h), (\mathrm{X}_d)\right) \,||\, \forall X. = \sum_{\text{value}}^{\text{limit}} X_{b^{dy}} == 0, \text{ No APDU}$$

The application layer bytes are limited and the remaining 56 bytes are employed for function $f(\mu_3)$. Such that: $f(\mu_3) = X_{b^{dy}}$, this function controls the bytes utilization within the whole stack, and contained bytes are dynamically employed for security storage purposes. The size of the dynamic buffer is increased and deceased according to the size of the application layer buffer such as single and multiple APDUs. However, the dynamic buffer size is enough during the deployment, and storage of security information and stack information [51].

Definition 4. The security development functions: $f(\mathrm{e}_1), f(\mathrm{e}_2), f(\mathrm{e}_3): f(Sym), f(Aym), f(H)$ are a hybrid function $f_{(E/D)^{Hy}}: (f_{E^{Hy}}, f_{D^{Hy}}) \in f(\mu_4) \in f_{\mu}$, which are deployed against stack vulnerability.

$$\forall X. f(\mathrm{e}_1, \mathrm{e}_2, \mathrm{e}_3)_{E[k]} \left\{ \forall X. \sum_{k=0}^{n} \left(f_{(\mu_1,\mu_2)}: (\mathrm{X}_h), (\mathrm{X}_d)\right) \right\} || \forall X. = \sum_{value}^{limit} X_{b^{dy}},$$
$$f(\mathrm{e}_1), f(\mathrm{e}_2), f(\mathrm{e}_3) \in f_{E^{Hy}} \in f(\mu_4) \in f_{\mu}$$

$$\forall X. f(\mathrm{e}_1, \mathrm{e}_2, \mathrm{e}_3)_{D[k]} \left\{ \forall X. f(\mathrm{e}_1, \mathrm{e}_2, \mathrm{e}_3)_{E[k]} (\forall X. \sum_{k=0}^{n} \left(f_{(\mu_1,\mu_2)}: (\mathrm{X}_h), (\mathrm{X}_d)\right) \right\} || \forall X. = \sum_{value}^{limit} X_{b^{dy}},$$
$$f(\mathrm{e}_1), f(\mathrm{e}_2), f(\mathrm{e}_3) \in f_{D^{Hy}} \in f(\mu_4) \in f_{\mu}$$

The security development is proportional to the communication requirements. Therefore, symmetric (*Sym*), asymmetric (Aym) (as an optional function) and hashing (H) are employed. At the other side, the dynamic buffer has sufficient space during the deployment of hybrid function.

Definition 5 (Bytes alignment). The number "*n*" bytes are aligned with lower layer bytes. The function $f(\mu_5) \in f_{\mu}$ is employed to form alignments between the bytes from the upper layer to the lower layer and *vice versa*.

$$\Leftrightarrow X \leq \mathrm{X}_d \geq \mathrm{X}_h \wedge (\mathrm{h}, \mathrm{d}) \leq \text{Limit}$$

$$\Rightarrow \exists f(\mu_5):$$

$$\forall X. \sum_{k=0}^{n} \left(f_{(\mu_1,\mu_2)}: (\mathrm{X}_h), (\mathrm{X}_d)\right) \xrightarrow[Alignment]{} \forall Y. \sum_{k=0}^{\max} \left(f_{(\mu_1,\mu_2)}: (\mathrm{Y}_h), (\mathrm{Y}_d)\right) \,||\, \forall XY.$$

$$\Rightarrow \sum_{\text{value}}^{\text{limit}} XY_{b^{dy}} \wedge \max \leq \text{Limit} \geq n$$

The "*n*" bytes are received from the upper layer and the function $f(\mu_5)$ stand between these bytes to align with lower layer bytes. If APDU is constructed and total size is limited as the protocol specified, then APDU bytes are fully aligned with lower layer bytes, including the last pseudo-transport layer block. The rest of the layers remain the same, without any change.

Definition 6 (Security, Bytes Reassemble). The function $f(\mu_6) \in f_{\mu}$ is employed to reassemble the bytes from the lower layer to the upper layer with security development.

$$\Leftrightarrow \exists[(\mathrm{X}_h, \mathrm{X}_d); (\mathrm{Y}_h, \mathrm{Y}_d); (\mathrm{Z}_h, \mathrm{Z}_d)] \Rightarrow f(\mu_6): (\mathrm{X}_h, \mathrm{X}_d) \underset{Ass/ReAss}{\Longleftrightarrow} (\mathrm{Y}_h, \mathrm{Y}_d) \underset{Ass/ReAss}{\Longleftrightarrow} (\mathrm{Z}_h, \mathrm{Z}_d)$$

The "n" bytes are transmitted from the upper layer to the lower layer; these bytes are assembled (Ass) according to lower layer requirements for further processing. On the other side, the bytes are received from the lower layer to the upper layer; these bytes are reassembled ($ReAss$) toward the user application layer.

To finalize and validate the security design, several implicit functions are employed from DNP3 sources (or from DNP3 open library), and explicit functions are deployed and encapsulated. To achieve this, formal postulates are employed that prove the proposed design and development. In these postulates, we use a dynamic cryptography buffer (DCB) as dynamic buffer (dB).

Postulate 1. Processing and computation of payload within the protocol (stack) flow (valuesEncrypt vE, numBytes nB, stackFlow sF, dynamicBuffer dB, offsets oS).

Determine the payload "P", the preceding bytes "B", the cryptography value and dynamicBuffer "dB" corresponding to manipulate bytes "mB" within the stack.

$$\sum_{k=0}^{\text{Limit}} \text{X}_{(n,dB)}\ , n = k, 1,2,3, \ldots., \text{limit} - 1, \text{limit} \qquad \text{Definition (1)}$$

The number of bytes "B" initialized with the initialization of dynamicBuffer "dB". While at the start, the offsets of the stack and "dB" are employed with empty bytes (cells) in the designed fields. The keyword "limit" specified the range of each field, and occupied bytes within the whole stack.

$$App_j.Comp\big(\text{oH}_{(n,dB)}, \text{dO}_{(n,dB)}\big)^{limit} \Rightarrow oH.Val(\text{O[bytes]}, \text{Q[bytes]}, \text{R[bytes]}) \wedge O.Val(\text{oG[bytes]}, \text{oV[bytes]}) \qquad \text{Definition (2)}$$

The user bytes are received and further treated to determine the application service data unit (ASDU) bytes. The ASDU bytes are computed by computing the objectHeader (oH) and dataObject ("dO": user defined name). During computing, the size of ASDU is limited up to 1990/1888 bytes, in the case of sending/responding bytes.

$$App_i.Comp\big(aH_{(n,dB)}\big)^{limit} \Longrightarrow aH.Val(\text{aC[bytes]}, \text{fC[bytes]} \vee aH.Val(\text{aC[bytes]}, \text{fC[bytes]}, \text{iiN[bytes]}) \qquad \text{Definition (2)}$$

Application header (aH) is computed, with the computing value of application control "aC" and function code "fC", in case of bytes' transmission. An additional value is computed, and designated as the internal indication "iiN" during the response bytes.

$$if \begin{cases} Comp\big(App_{(i,j)}\big) \leq \text{limit}, \text{Single APDU} \\ Comp\big(App_{(i,j)}\big) \geq \text{limit}, \text{Multiple APDUs} \\ Comp\big(App_{(i,j)}\big) = 0, \text{Header bytes only.} \end{cases}$$

The application protocol data unit (APDU) is computed by computing the ASDU bytes, and header bytes that are also designated as application protocol control information (APCI). If $Comp\big(App_{(i,j)}\big) = 0$, only APCI information is transmitted, without any data (or information bytes).

Then,

$$\text{payLoad}_1.\text{Crypt}\{Comp(\ App_{(i,j)})\} \implies (Aym/Sym|Hash)\{Comp(\ App_{(i,j)})\} \\ \| \ dB.Crpto[bytes] \wedge dB.update[bytes]$$ Definition (3; 4)

The payLoad_1 is computed by deploying the cryptography function (or algorithms); asymmetric and symmetric functions are optional and used according to communication demands. The buffer employed 56 bytes from the application layer and the available security information is updated correspondingly in dynamicBuffer "dB". The buffer keeps track of security deployment and related information about bytes from the stack [51].

$$if \begin{cases} \text{payLoad}_1 \leq \text{limit} \geq dB.Crpto[bytes]\text{, Process information} \\ \text{payLoad}_1 \leq \text{limit} \leq dB.Crpto[bytes]\text{, Memory Initialization} \\ \text{payLoad}_1; dB.Crpto[bytes] = 0\text{, No security function.} \end{cases}$$

Then, during security development, the buffer is full (indicated by exception). Then, additional memory has been located from the data link layer stack. A CRC field occupied 34 bytes in the data link layer stack. In case bytes are transmitted without implementation of security or security functions are not available then, $dB.Crpto[bytes] = 0$, which shows that the payload is not manipulated with security (functions).

Postulate 2. Determine and disassemble the bytes with alignment, and compute the function for further security deployment (EncryptDecryptValues V, numBytes nB, stackFlowBytes FB, dynamicBuffer dB, dynamicBufferFlow BF, offsets oS).

Suppose "Y" is the sum of bytes received from upper layer. The application layer (level) stack has been filled with number of bytes "B" and dynamicBuffer "dB" is updated with corresponding information. Therefore, "dB" occupied space, while the offsets in the pseudo-transport layer stack are empty bytes (cells) in designed fields during initialization.

$$\Leftrightarrow \sum_{k=0}^{Limit} \text{Y}_{(n,dB)} \implies \left(Y_{Tr_j} + Y_{Tr_i}\right)^n = \sum_{k=0}^{n=limit} \binom{n}{k} {Y_{Tr_j}}^k {Y_{Tr_i}}^{n-k}$$ Definition (5; 6)

The bytes are being aligned and disassembled into pseudo-transport layer stack, with specifications in mind. The upper layer bytes are disassembled in to fixed blocks, and each block has 249 bytes with one byte of header field or transport protocol control information (TPCI). Each transport protocol data unit (TPDU) size is limited up to 250 bytes, which included the meaningful data (from upper layer) and header bytes.

$$if \begin{cases} Comp\left(Y_{Tr_j} + Y_{Tr_i}\right) \neq 0\text{, Single TPDU} \\ Comp\left(Y_{Tr_j} + Y_{Tr_i}\right)^n \geq \text{limit, Multiple TPDUs} \\ Comp\left(Y_{Tr_j} + Y_{Tr_i}\right) = 0\text{, Header bytes only.} \end{cases}$$

Then,

$$\text{payLoad}_2.\text{Crypt}\left\{Comp\left(\ Y_{Tr_{(i,j)}}\right)\right\} \implies (Hash)\left\{Comp\left(\ Y_{Tr_{(i,j)}}\right)\right\} \\ \| \ dB.Flow[bytes] \ \wedge dB.Crpto[bytes] \ \wedge dB.update[bytes]$$ Definition (3; 4)

The TPDU bytes are constructed and security is deployed and computed using the hashing function or algorithm. At this stage, corresponding information is updated in the buffer. The bytes' flow has been verified to ensure the sufficient memory space as well as tracking of each layer of bytes and deployed security information.

$$if \begin{cases} \text{payLoad}_2 \leq \text{limit} \geq dB.Hash[bytes]|| dB.Flow[bytes], \text{Process information} \\ \text{payLoad}_2 \leq \text{limit} \leq dB.Hash[bytes]||dB.Flow[bytes], \text{Memory Initialized} \\ \text{payLoad}_2; dB.Hash[bytes]||dB.Flow[bytes] = 0, \text{No security function.} \end{cases}$$

The security function using hashing is deployed on TPDU bytes if the buffer size is sufficient. Otherwise, additional space is initialized from the link layer. The "0" indicated that security function is not available, after processing of TPDU bytes. In case, if only header bytes have been received from upper layer then, TPCI is transmitted with a security layer check.

Postulate 3. The bytes are assembled/reassembled corresponding with link layer frame, and the security function is computed within proprietary stack (valuesEncryptDecryptValues V, numBytes nB, assembledBytes aB, reassembledBytes rB, stackFlowBytes FB, dynamicBufferFlow BF, dynamicUpdateBuffer UB, offsets oS).

The bytes "Z" are received from the upper layer and are assembled. The stack fields have been updated with upcoming bytes and corresponding information. The situated updated bytes and related flow of information in dynamicBuffer "dB" have been checked and a special field designated as "dynamic flow checker" is used, which follows the flow of bytes in buffer and shows the indication/exception when buffer memory is full.

$$Z_{Dl_j}.Comp(Z_i + Z_j)^{n=limit}, Z_i \wedge Z_j \notin Z \bigwedge \in Z_{Dl_i} \qquad \text{Definition (2)}$$

Here, "limit" shows the number of blocks employed during processing of thw frame.

The upper layer bytes are employed and expended into 16 blocks (Z_i: assembled bytes having 250 bytes) with CRC (Z_j: optional field having 32 bytes). Each block has limited size, while final block is processed with 10 bytes.

$$Z_{Dl_i} \Longrightarrow Z_{Dl_i}.Val(\text{S[bytes], L[bytes], C[bytes], DA[bytes], CRC[bytes]}) \qquad \text{Definition (2)}$$

The value of Z_{Dl_i} (having subfields: two bytes of start "S"; 1 byte of length "L", one byte of control "C", two bytes of destination address "DA" and two bytes of cyclic redundancy check "CRC") is added with Z_{Dlj}, which made the complete link layer frame. In Z_{Dli}, the CRC function has been employed, but in case of Z_{Dlj}, this function is treated as an optional field.

$$\Longrightarrow \left(Z_{Dl_j} + Z_{Dl_i}\right)^n = \sum_{k=0}^{n=limit} \binom{n}{k} {Z_{Dl_j}}^k {Z_{Dl_i}}^{n-k} \qquad \text{Definition (5; 6)}$$

$$if \begin{cases} Comp\left(Z_{Dl_j} + Z_{Dl_i}\right) \neq 0, \text{Single LPDU} \\ Comp\left(Z_{Dl_j} + Z_{Dl_i}\right)^n \geq \text{limit}, \text{Multiple LPDUs} \\ Comp\left(Z_{Dl_j} + Z_{Dl_i}\right) = 0, \text{Header bytes only.} \end{cases}$$

The disassembling/reassembling process from transport level made the bytes convenient for the data link frame. In case of "0", link protocol control information (LPCI) bytes are transmitted, rather than the complete frame or LPDU.

Then,

$$\text{payLoad}_3.\text{Crypt}\left\{Comp\left(Z_{Dl(i,j)}\right)\right\} \Longrightarrow (Aym/Sym|Hash)\left\{Comp\left(Z_{Dl(i,j)}\right)\right\} \\ \| \, dB.Flow[bytes] \ \wedge dB.Crpto[bytes] \ \wedge dB.update[bytes] \quad \text{Definition (3; 4)}$$

The security has been deployed with two optional functions designated as payLoad_3. A few experimental tests observed that the deployed security is inappropriate for LPCI bytes at the data link layer. Because LPCI has source and destination fields, if these fields have encrypted values, then this will be complex to compute at the destination side during the sender/receiver identification process. Therefore, LPCI bytes are not encrypted; only the hash value is generated to verify the bytes' integrity. In this case, encrypting the LPCI as part of a LPDU would be required, and external (or additional) source and destination addresses are added (as explained in Section 5).

The bytes' flow has been checked and, simultaneously, security information is updated within the buffer. This would be further employed during verification of security or the decryption process at the destination. During measurement, a few experimental tests concluded that the buffer is full, and required additional memory space. Therefore, this exception has been resolved by utilization of 32 byte CRC from the data link layer.

$$if \begin{cases} \text{payLoad}_3 \ \leq \text{limit} \geq \ dB.Hash[bytes]||dB.Flow[bytes] \text{ , Process information} \\ \text{payLoad}_3 \ \leq \text{limit} \leq dB.Hash[bytes]||dB.Flow[bytes] \text{ , Memory Initialized} \\ \text{payLoad}_3 \text{ ; } dB \cdot Hash[bytes]||dB.Flow[bytes] = 0 \text{ , No security function.} \end{cases}$$

Then, if the payLoad_3 is limited up to 260 bytes and the buffer is updated with security information, then the desired bytes are processed to the physical layer. Otherwise, extra memory would be occupied by a buffer that conveniently provided the security information for a security level check.

Postulate 4. Compute, compile and verify the security (totalPayload tP, checkSecurity cS, encryptdecryptValues V, numBytes nB, assembledBytes aB, reassembledBytes rB, stackFlowBytes FB, dynamicBufferFlow BF, dynamicUpdateBuffer UB, offsets oS).

The $f(\text{tP})$ is an explicit function that combines and manipulates the security at each layer and retrieves the desired information from the buffer.

$$f(\text{tP}) = \text{Val}\{\text{payLoad}_1; \text{payLoad}_2; \ \text{payLoad}_3\} \quad \text{Definition (3; 4; 6)}$$

$$\Longrightarrow \text{Decrypt}[f(\text{tP})] \, \| \, dB.Flow[bytes] \ \wedge dB.Crpto[bytes] \ \wedge dB.update[bytes]$$

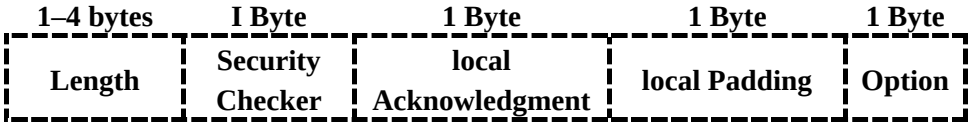
The security has been deployed and integrated as an additional layer (or three-layer security) with desired layers within the stack. At the destination, the security using cryptography has been validated and verified, before transmitting bytes to the upper layer.

5. Protocol Bytes and DCB

The application layer is the most sensitive layer compared with other layers of DNP3 because it generates and distinguishes between messages at that layer [3,23,32]. The original size of the application layer fragment is 2048 bytes, which is also specified by original DNP3 documentation [5]. In the proposed study, the application protocol data unit (APDU) size is limited up to 1992 bytes, and the remaining 56 bytes are employed to deploy the DCB [51]. Moreover, Figure 6 shows the DCB bytes against APDU bytes during security development. The DCB is a dynamic bytes buffer, which dynamically stores and tracks DNP3 generated bytes and related information. Dynamic fields are deployed in DCB, and the "Bytes Selected" field is used to control and track the original DNP3 manipulated bytes. Further details related to each field of DCB are illustrated in Figure 7 and explained as follows:

User Bytes: Bytes Selected: This is the dynamic length field in DCB which keeps the information about protocol manipulated bytes and the security implementation bytes. Dynamic means that the additional bytes are used from dynamic storage, depending on requirements. This field is composed of five main subfields: length, security checker, local acknowledgment, local padding and optional.

This field is individually employed to perform the relative functions and to keep and monitor the information on the application layer development. Then, this field is ready to measure the information of the lower layer(s). In the application layer, APDU (with security) bytes are computed and the remaining memory is dynamically allocated to DCB, which would be further utilized. However, we have not conceded if the additional memory is available in TPDU and LPDU buffers. The remaining bytes are treated as padding bytes, which ensure that the development ends; the shaded area in each buffer including the APDU buffer, TPDU buffer and LPDU buffer in Figure 4 represents the padding bytes.



Initially, eight bytes are occupied by this field, but this number would be increased according to the requirements.

Length: Count of protocol constructed bytes and security development bytes. The range is 0–65,535, which would be convenient if maximum bytes are received by the application layer. Basically, this range is allocated individually to count the bytes, and the dynamic assigned to lower layer(s) after completion of the upper layer length (or bytes). This means that first APDU bytes are counted. Then, the range is defined for the lower layer(s).

Security Checker: Ensures that security has been deployed in each layer and checks which algorithm(s) are used during development.

Local Acknowledgment: This is an exceptional message followed by security checker status.

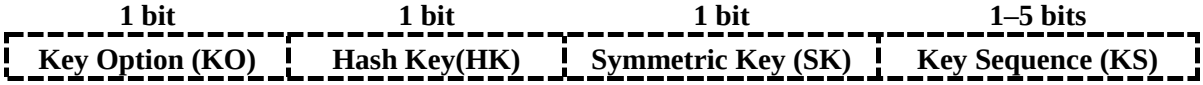
Local Padding: The remaining bytes are padded, and status is shifted to dynamic padding (field) in DCB.

Option: This field contains one byte, which is significant. It keeps information, such as data type, maximum APDU size, polling and timestamp.

Cryptography Key Sequence

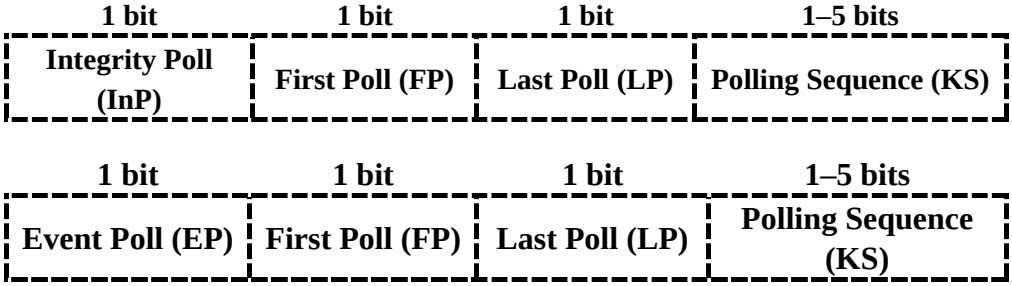
This is a one-byte field, which keeps the information about cryptography keys, and it is employed in security development. A single byte of "cryptography key sequence" is distributed as: two bits identify the first key and last key that are employed in security development, one bit identifies the key option that performs a special security function on demand, one bit is used to select the security method and four bits act as a key sequence counter.

In security development, two bits are used to select the cryptography method(s), such as hashing and symmetric methods, and then corresponding information is collected. The key option subfield occupied one bit that is designated as the optional algorithm used from the arena of cryptography. The hash (function) digest is calculated, defined as the hash key and added into the key sequence counter.



Polling Sequence

Two bytes are defined for integrity and event polling in DCB. In polling, one bit indicates initiating the polling: integrity and event, two bits are used to identify the first and last polls, after initialization of polling and a five bits polling counter is added in a sequence of 0–31. In the proposed study, the sequence 0–31 is used during integrity polling, and the sequence is changed as 32–63 in event polling. During polling, integrity/event polls can start from a value within specified ranges and increment the polls in sequences thereafter. The sequences roll as 31–0 and 63–32 during a response or/and decryption process.



Cryptography: Dynamic Storage

This is a special field, which contains variable bytes of DCB. At the initialization stage, this field contains 20–56 bytes, and would be changed to dynamic. For example, event polling is not performed. Thus, the field occupied bytes are shifted to dynamic storage. On the other side, the bytes are dynamically shifted to the polling sequence in case additional bytes are required by event polling.

Dynamic Padding

This field is part of dynamic storage. APDU bytes are constructed with security development and remaining bytes are padded and then added in DCB. This field may inherit the padding operational bytes of local padded from "User Bytes: Bytes Selected".

Optional

This field contains one byte, which verifies the content of polling, before transmitting it to the recipient(s).

Non-Critical

A one-byte field that shows normal polling status in SCADA/DNP3 transmission. This byte travels along polling, and analyzes the communication status.

Critical

This field also contains one-byte information, travels along polling and analyzes the abnormal (or critical) communication status. In terms of analysis, one to two bits are enough to analyze the normal or abnormal scenario of polling, but one byte is occupied by each field including critical and non-critical, and the remaining bits are reserved for future development.

Solution: Select Method

This field contains one byte, which identifies the methods (or cryptography methods) currently employed in security development. This field may inherit the approach of "security checker" (subfield in "User Bytes: Bytes Selected"), but performance is limited. This field performs additional functions compared to "security checker" performance. For example, abnormal transmission is observed by a "critical byte" and non-repudiation function is required to perform. Therefore, a digital signature is accounted for by this field (or "solution: select method"). This type of security is performed at the time of initialization (or polling initialization).

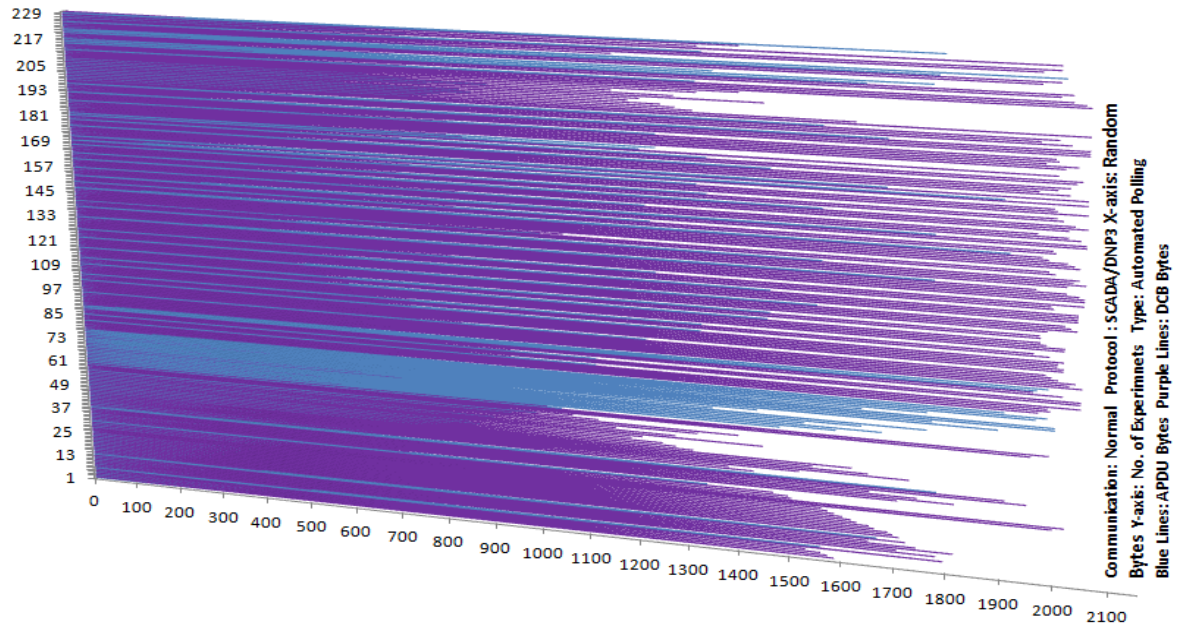


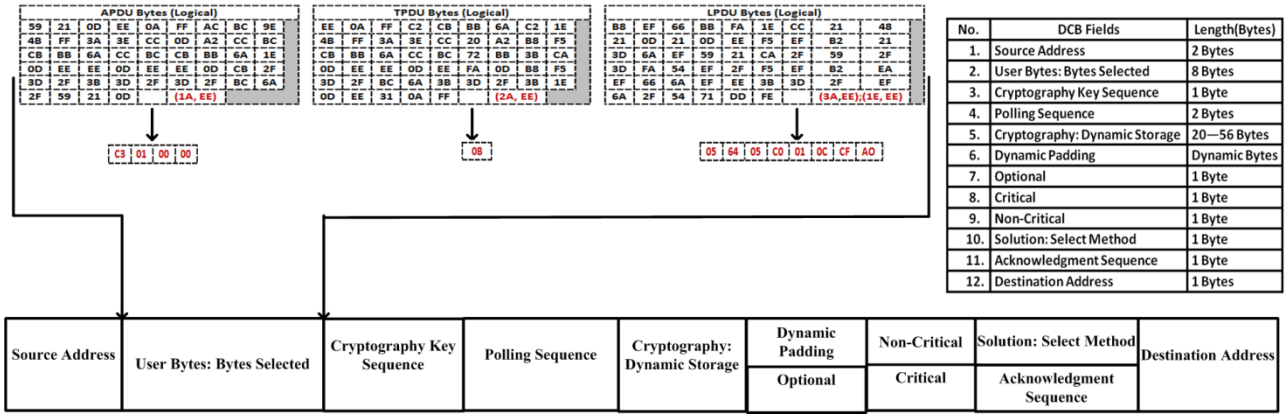
Figure 6. DCB bytes against APDU bytes during security development: The blue lines show the computed APDU bytes along the y-axis and the remaining bytes visualized in purple are employed in DCB, plus the original 56 bytes. The bytes are constructed in the application layer stack and the remaining bytes are padded and further assigned to DCB, which ensures the completion of the APDU process.

Acknowledgment Sequence

This field contains one byte and is usually employed by the main controller. During polling, the response has been transmitted from the sub-controller, and the main controller will reply with acknowledgment corresponding to polls (or integrity and event polls), while six bits are employed in that case. The remaining two bits are employed during the time of polling initialization.

Source and Destination Addresses

Four bytes are allocated for source and destination addresses so that each one has two bytes each. When a frame (or LPDU) is ready to transmit to open protocols (or TCP/IP) the additional source and destination addresses are added. Because, in a few cases, the receiver cannot identify the addresses due to security development (or encryption), additional source/destination addresses are added with an encrypted message. It will also be worthwhile to compare the addresses with link layer specified source/destination addresses after decryption.



No.	DCB Fields	Length(Bytes)
1.	Source Address	2 Bytes
2.	User Bytes: Bytes Selected	8 Bytes
3.	Cryptography Key Sequence	1 Byte
4.	Polling Sequence	2 Bytes
5.	Cryptography: Dynamic Storage	20—56 Bytes
6.	Dynamic Padding	Dynamic Bytes
7.	Optional	1 Byte
8.	Critical	1 Byte
9.	Non-Critical	1 Byte
10.	Solution: Select Method	1 Byte
11.	Acknowledgment Sequence	1 Byte
12.	Destination Address	1 Bytes

Figure 7. The DCB contained numbers of fields, and each field occupied specific/dynamic space to retain the information during development. The desired protocol bytes are generated, and represented in a hexadecimal notation. The empty cells with corresponding solid arrows in message blocks, including the APDU block, TPDU block, and LPDU block, represent the header fields, and arrows are directed toward header bytes. The red bytes at each block represent the security implementation bytes.

6. Testbed Setup

In the automated SCADA/DNP3 testbed, eight remote terminal units (RTUs) or sub-controllers are configured with the mater terminal unit (MTU) or main controller, with the bandwidth of 5 Mbps. Each remote station is connected directly with a water pumping system, and designed to collect information from the system through sensors. At the start, the main controller initiates the communication, and remote station(s) will reply with a message with simple status information, acknowledgement and polling information.

Preliminarily, three intervals are specified for automated polling. These are intervals of four, seven and 10 min, but afterwards the interval time will be increased. Performance is observed in both cases of the testbed including normal transmission (without attack scenario) and abnormal transmission (with

attack scenario) in distinguished time variations to validate the security development. Figure 8 shows the simulation environment of where the experiments and measurements are conducted [35].

In automated polling, the sub-controller will send the integrity poll response every 45 s and a normal event is transmitted every 20 s at specified intervals. If we divide the four-minute interval, integrity polls will respond five times to the main controller and the remaining 15 s remain unused or a dummy poll, such as an exception message, is transmitted. Frequently, 12 normal events are generated which show the transmission status followed by critical/non-critical functions, while the abnormal event will be transmitted any time and may be in the form of report-by-exception or changes accounted for in the last integrity poll.

For example, we have initialized the interval for automated polling, in which the integrity poll occurred every 45 s and normal event poll occurred every 20 s, after the initiation of the transmission. In automated polling, every time integrity and normal event polls occur, they are counted in distinct sequence numbers as 0–31 and 32–63. More details are illustrated in figure 8.

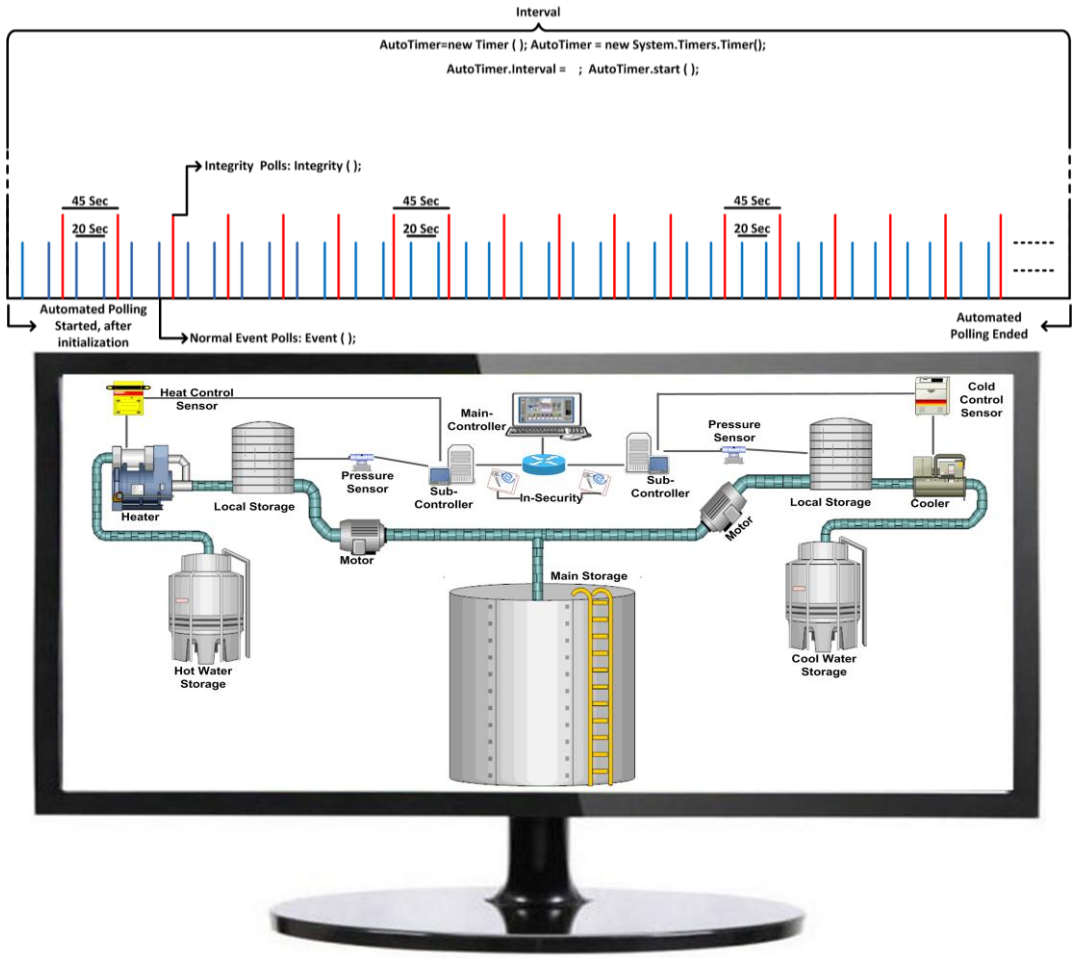


Figure 8. Simulation design and environment.

In the section below, the number of logical states is defined which show the overall development. Overall development includes the number of integrity/event polls in automated polling corresponding to intervals, security development during automated polling, normal/abnormal transmission, and security achievements (or proposed security achievements).

7. Automated Polling Design and Flow

Here, we first divide the security development into two phases: S-Solution[1] and S-Solution[2]. S-Solution[1] follows the same method of Section 4, while the S-Solution[2] also computes a digital signature at the data link layer to verify the non-repudiation security [8,19]. At the start, the master node initiates the automated polling and a confirmation message is sent from the sub-controller(s). However, the S-Solution[2] is deployed at the time of initialization.

In automated polling, the sub-controller (or RTU) should send an integrity poll response every 45 s, and a normal event poll is transmitted every 20 s at specified intervals of four, seven and 10 min. The abnormal event poll will be transmitted without following the specific time. However, the S-Solution[2] is deployed during polling.

During the initialization process, and acknowledgement from the sub-controller, the message also contained all requirements parameters including interval, number of integrity and event poll occurrences at the specified interval, security solution, and other requirements. A special explicit field designated as "solution: select method" has been employed in DCB which should identify security development, such as the S-Solution[1] and S-Solution[2]. This field also changes the status of security development, by analyzing the level of security indicated or measured from the critical functional field.

In Figure 9, several states are used which designate the communication flow that occurs in automated polling. The state 803 shows the master node encryption process, while state 804 and state 805 show the remote node decryption process, and state 806 for acknowledgement. At state 807, the remote node sends the integrity poll response continuously to the master station every 45 s, and a normal event is transmitted every 20 s at specified intervals. The sub-states included state (807, 0_0) __ (S.., St..._ Fn…) of state 807 represent the response integrity polls against the master controller request (or polling request). The state 808, state 809 and onward states represent the remote node encryption process, while these states are optional; the state 810, state 811 and onward states represent the master node decryption process; and forward states show that communication has been continued. More detail of automated polling in normal transmission is depicted in Table 2.

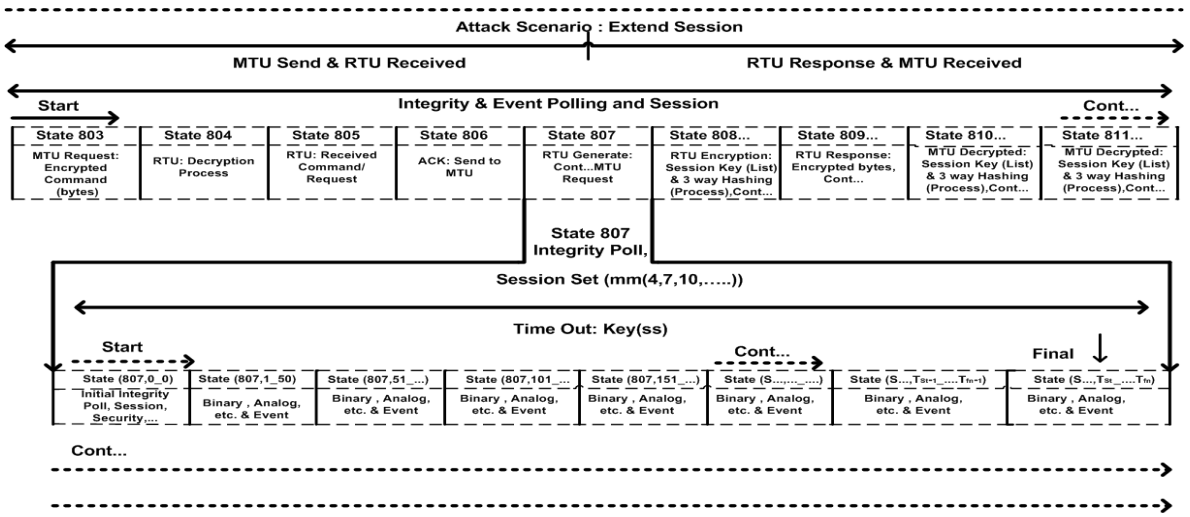


Figure 9. Automated polling in normal communication.

Table 2. Logical states: automated polling in normal communication.

State (Logical)	Process	Security Via
803	Master station initiates the communication with encryption (bytes).	Security Via
804, 805	Remote station decrypted the message (bytes) and bytes received.	S-Solution^2
806	Acknowledgement: From RTU to MTU.	S-Solution^2
807	The Integrity Poll: Remote station generates continuously response.	S-Solution^1
(807, 0_0) __	The Integrity Poll: Generate continuously response.	S-Solution^1
808…, 809,…	RTU: Encryption process.	S-Solution^1
810…, 811,…	MTU: Decryption process.	S-Solution^1

This is a simulation-based work, and developed in C#; to check incoming/outgoing water flow to/from water tank, sensors reading points, and status. At the start, static points are slotted at the passage of time in automated polling; the points are interchanged with abnormal points, which interrupt the normal sequence of polling. The normal event occurred every 20 s which usually shows configuration and connectivity status, points' status, session key time out (optional) and security status, followed by DCB fields. Meanwhile, an abnormal event will occur at any time if significant change is observed from field devices; for example, the tank water level increased from its normal flow, the power consumption point suddenly increased, the device status changed to unknown, and the detection of an unknown entity occurred in the automated polling channel.

In some cases (or critical cases), when the main controller or remote node wants to authenticate each other, or unknown entities are continuously interrupting automated polling, then the S-Solution^2 is employed to replace the S-Solution^1. Typically, S-Solution^2 utilization is time consuming because of the asymmetric encryption (*i.e.*, RSA algorithm) but security is successfully achieved, even in the case of critical/abnormal polling because the digital signature is computed and this signature evaluates the non-repudiation security in SCADA transmission.

In Figure 10, automated polling has been continued at state 906 to state 909. The state A0, state A11, and state A30 show the abnormal communication. This means that attacks, such as authentication, integrity, and confidentiality attacks, are successfully launched, detected and the SCADA/DNP3 main controller is unaware of these during this abnormal scenario. The abnormal scenario is designed to measure the impact level of the attacks in automated polling, as a part of SCADA/DNP3 system and to validate the security development. A field called "critical" is used to identify the abnormal communication. If abnormal transmission has continuously occurred, then an abnormal event is transmitted from the main controller to the remote terminal unit (RTU) by the employment of the S-Solution^2. At state A41, the non-repudiation function (or security) is tested between SCADA/DNP3 nodes.

Figure 11 shows that the attacks, such as authentication, integrity, and confidentiality attacks, are successfully launched. They are detected at state A207, state A312, and state A375, and the SCADA/DNP3 remote node is unaware during abnormal communication. The sub states, state (A207, 0_0) to state (S..., $T_{St\_}$....T_{fn}) of state A207; state (A312, 0_0)__ state (S..., $T_{St\_}$....T_{fn}) of state A312; and state (A375, 0_0) to state (S..., $T_{St\ \_}$....T_{fn}) of state A375,represent the polling (or communication sequence). They also indicate the status of the security level with an optional function that can be employed against non-repudiation attacks. More detail related to attacks' detection at the main-controller/sub-controller(s) sides and the related communication sequence are depicted in Table 3.

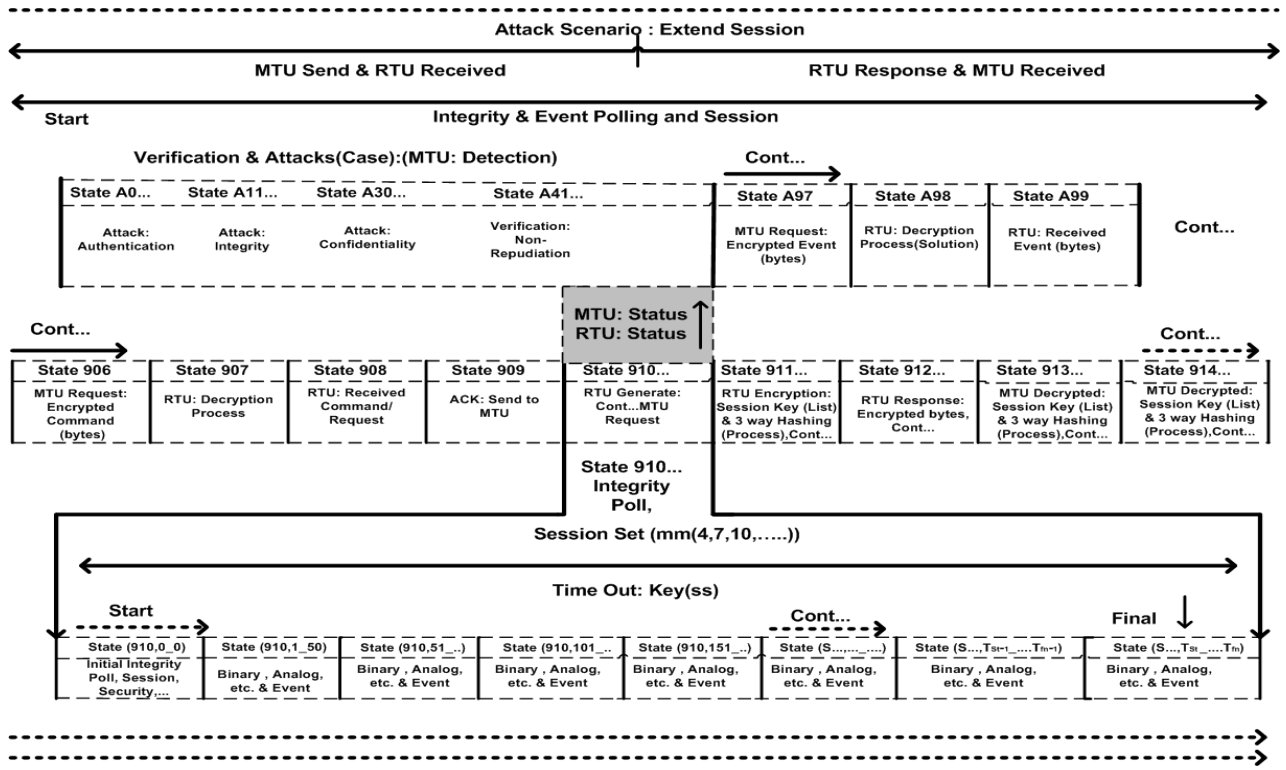


Figure 10. Automated Polling in abnormal communication at main controller side.

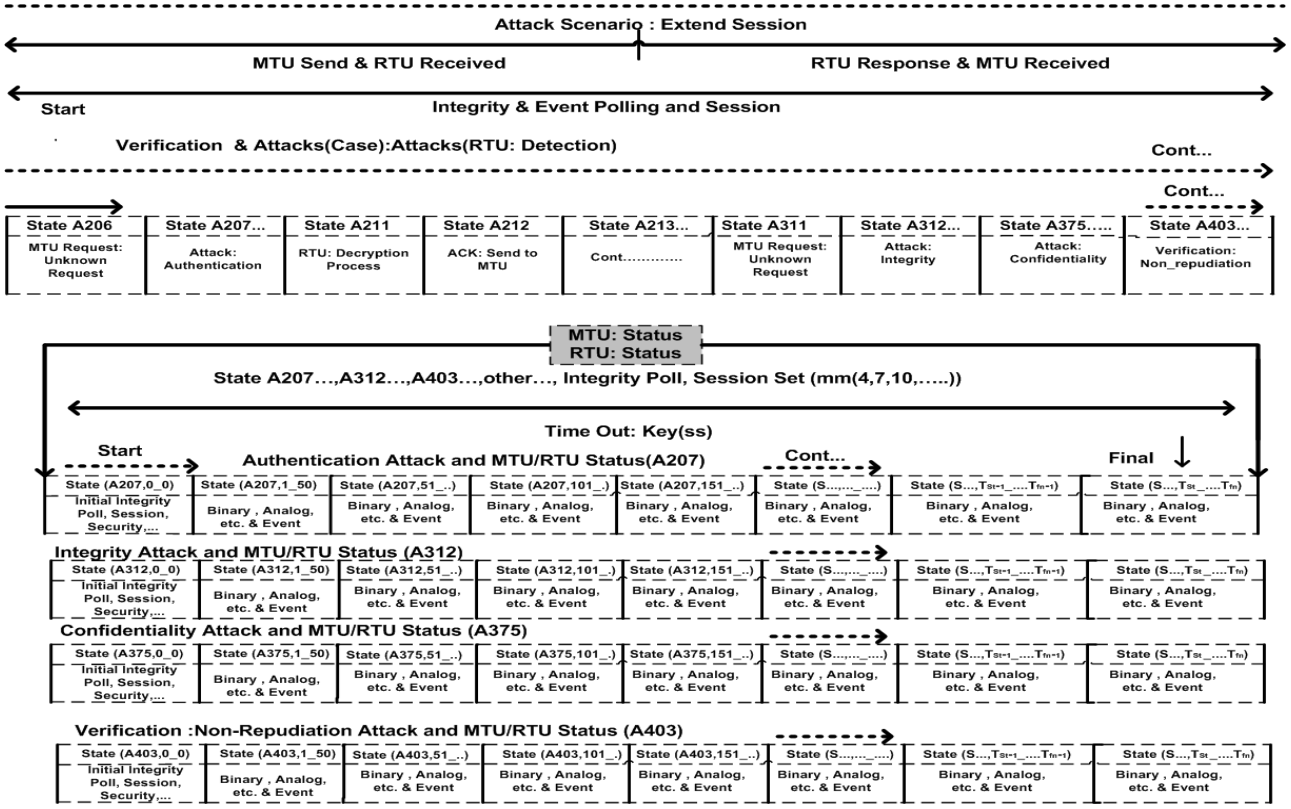


Figure 11. Automated polling in abnormal communication at the sub-controller(s) side.

Table 3. Logical states: automated polling in abnormal communication.

MTU: State (Logical)	MTU: Process	RTU: State (Logical)	RTU: Process
906	MTU Request: Encryption process (bytes).	A206	Unknown bytes.
907	RTU Decryption process (bytes).	A207	Authentication Attacks: Verification.
908	RTU has received MTU request.	A211	RTU: Decryption process.
909	Acknowledgement.	A212	Acknowledgement.
910	The Integrity Poll: Remote station generates continuously response.	(A207, 0_0)__	Generate continuous response: Verification process
(910, 0_0)__	The Integrity Poll: Generate continuously response.	A213	Continuous response
A0	Abnormal Communication: Detection of Authentication Attacks.	A311	Unknown bytes.
A11	Abnormal Communication: Detection of Confidentiality Attacks.	A312	Integrity Attacks: Verification and Continuous
A30	Abnormal Communication: Detection of Integrity Attacks.	(A312, 0_0)__	Generate continuous response: Verification process.
A41	Abnormal Communication: Detection of Non-Repudiation Attacks (Optional).	A375	Confidentiality Attacks: Verification and continuous
A97	MTU Request: Event (in-case, RTU does not reply or in Abnormal scenario).	(A375, 0_0)__	Generate continuous response: Verification process.
A98	RTU Decryption: Event (Incase, RTU does not reply or Abnormal scenario).	A403	Non-Repudiation Attacks: Verification and continuous.
A99	RTU has received MTU request and process continuous…….	(A403, 0_0)__	Generate continuous response: Verification process.
911…, 912,..	RTU: Encryption process	–	–
913…, 914,..	MTU: Decryption process	–	–

8. Performance Measurement and Discussion

In this section, performance results are measured to validate the security development. To validate the security of automated polling, attack scenarios have been designed to act as un-authorized entities in communication. This study aims to achieve the security functions including authentication, integrity and confidentiality, as well as non-repudiation functions in a few critical cases. To achieve the security functions, related attacks are launched using built-in-tools, and are successful in warning and interrupting the automated polling. Table 4 depicts the number of tools that are used in abnormal scenarios and corresponding attacks.

To evaluate performance, we categorize and compute the results into three scenarios:

- In the first scenario, the attacks are launched but DNP3 model is designed without security;
- In the second scenario, the attacks are launched and the DNP3 model is designed with security development, and
- In the third scenario, security is deployed at each end of the automated polling, and performance is observed against attacks. These scenarios are also useful to compare the overall computed performances.

Table 4. Tools and related attacks.

Security Functions	Tools	Attacks
Authentication	Cracking Tools, Sniffer, Dsniff, Winsniffer and Password Dictionary	Guessing Shared Key, Brute Force and Password Guessing
Confidentiality	Ethereal, Ettercap, Kismet, Aircrack, Airsnort, Dsniff, and Ettercap	Eavesdropping, Key Cracking and Man-in-the-Middle
Integrity	Airpwn, File2air, Dinject/Reinject, Capture and Injection Tools, Jamming and Injection Tools	Frame Injection, Data Replay and Data Deletion

In each scenario in the testbed, successful experiments are performed 229 times and performance results are measured at specified intervals. Total attack detection and computed impact on automated polling act as a mirror to compute security performance results and the validation of development. The numbers of experiments show the most significant computed attack detection in automated polling and are helpful in computing the corresponding security.

In the testbed, the maximum size of the user payload is limited to 1992 bytes. This means that the APDU size is limited to 1992 bytes in security development. The integrity poll response is transmitted every 45 s, and the maximum size of the payload is 1992 bytes, plus security bytes. On the other side, the normal event is polled every 20 s, and the specified time is sufficient because a normal event is just a message or an exception message of a few bytes, such as the first or last integrity poll transmitted and normal polling status and other acknowledgement messages. The number of times random bytes are polled to measure the total time of polling, which is under 45 s in the case of integrity polls and 20 s in the case of normal event polls. To enhance the security within a specified time, two changes are made in the testbed. In the first change, the session is added with a symmetric key. Therefore, the lifetime of the keys is 45 s and 20 s in case of integrity and the normal event. However, the total interval and polling sessions would change according to the requirements. However, the interval is increased but polling sessions are fixed in the testbed. In the event that the numbers of bytes are increased from the maximum limit, the polling session will also be changed. In the second change, the symmetry key is used to perform encryption at a data link layer and a hashing function is employed on a symmetry key, not on LPDU bytes. The session time is also decreased and the symmetric key is protected from unknown entities.

For example, if an attacker has stolen the symmetric key and the hash digests match, the main controller computes the hash of the symmetric key shared securely among the participated nodes. This key is employed to perform encryption, otherwise bytes are rejected and a new poll will be transmitted from the sub-controller. Each poll (such as integrity and event poll) is counted by a polling sequence counter at both sides.

Attack Detection and Security

A total of 229 experiments are employed to compute the overall performance. In total, an initial experiment (or experiment No. 0) is tested to verify the testbed configuration, and connectivity between nodes; while experiment No. 26 and No. 17 of the second and third scenarios are tested for non-repudiation security verification.

In the first scenario, Figure 12 indicates that attacks are launched to interrupt the sensitive information of automated polling. In each experiment, attacks are launched and corresponding behaviors are measured. During attack detection, there are no security mechanisms, such as OS security, firewalls and others, which are configured to provide security in automated polling. The detected attacks are represented by color markers: red shows the authentication attacks, black shows the confidentiality attacks, and orange shows the integrity attacks. The blue lines show the random data rates along the *y*-axis corresponding to experiments in the *x*-axis, and green lines show automated polling flow as normal (or without attack case) and abnormal (or with attack case). As shown in Figure 12, large numbers of attacks are detected in absence of security mechanisms, which shows that DNP3 is designed to protect against potential vulnerabilities and attacks.

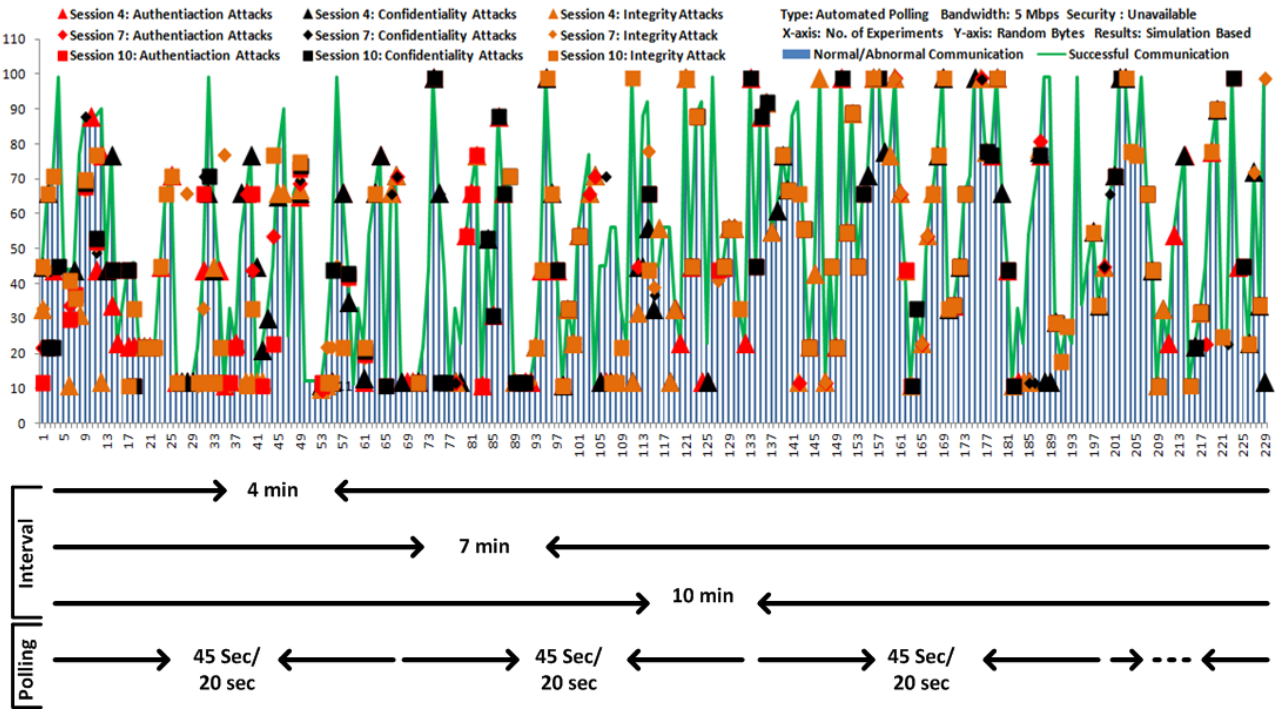


Figure 12. Automated polling (Scenario 1): Attack detection at specified intervals.

Similar experiments are repeated in Figures 13 and 14, which show attack detection in automated polling with security development. However, security design is changed in both performance tests, followed by the second scenario and third scenario. After computing the performance results of the first scenario, DNP3 bytes are polled without security concerns, while security has been developed in the second scenario and attack detection is measured in Figure 13. In the second scenario, security is deployed inside the DNP3 stack model, which is one of the main contributions of this study. In the third scenario, security development is made at each end of the poll or the message, and the total detected attacks are visualized in Figure 14.

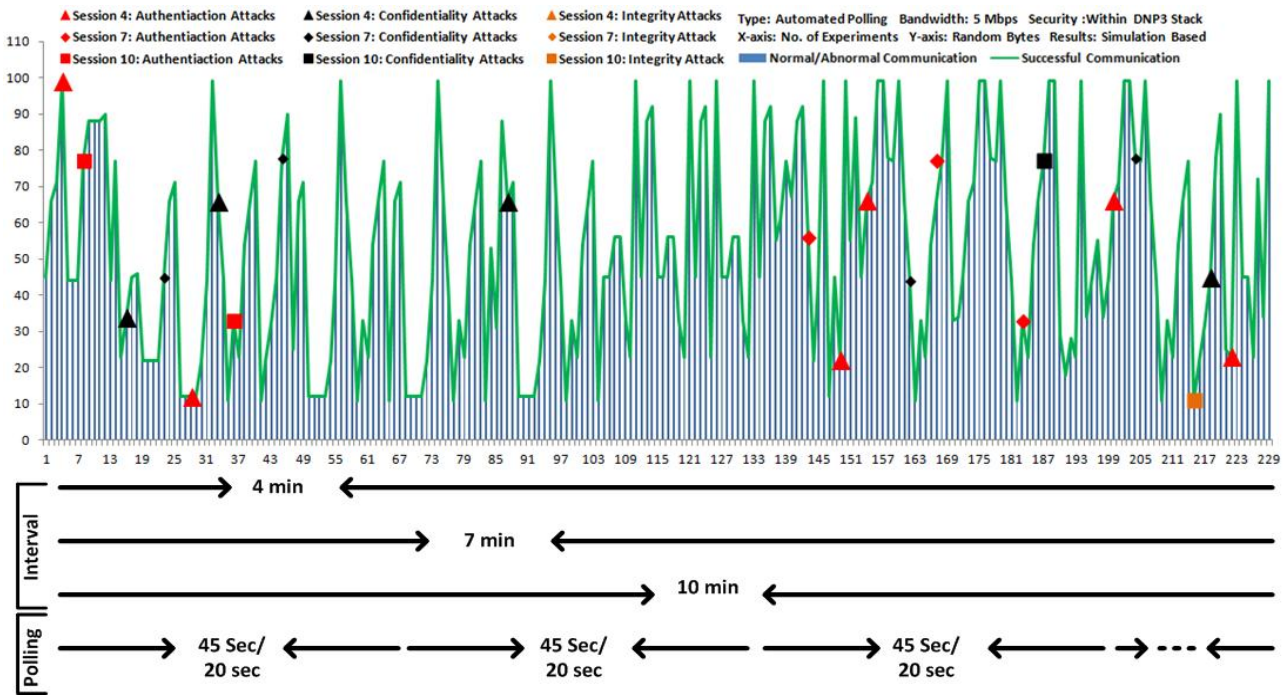


Figure 13. Automated polling (Scenario 2): Attack detection at specified intervals.

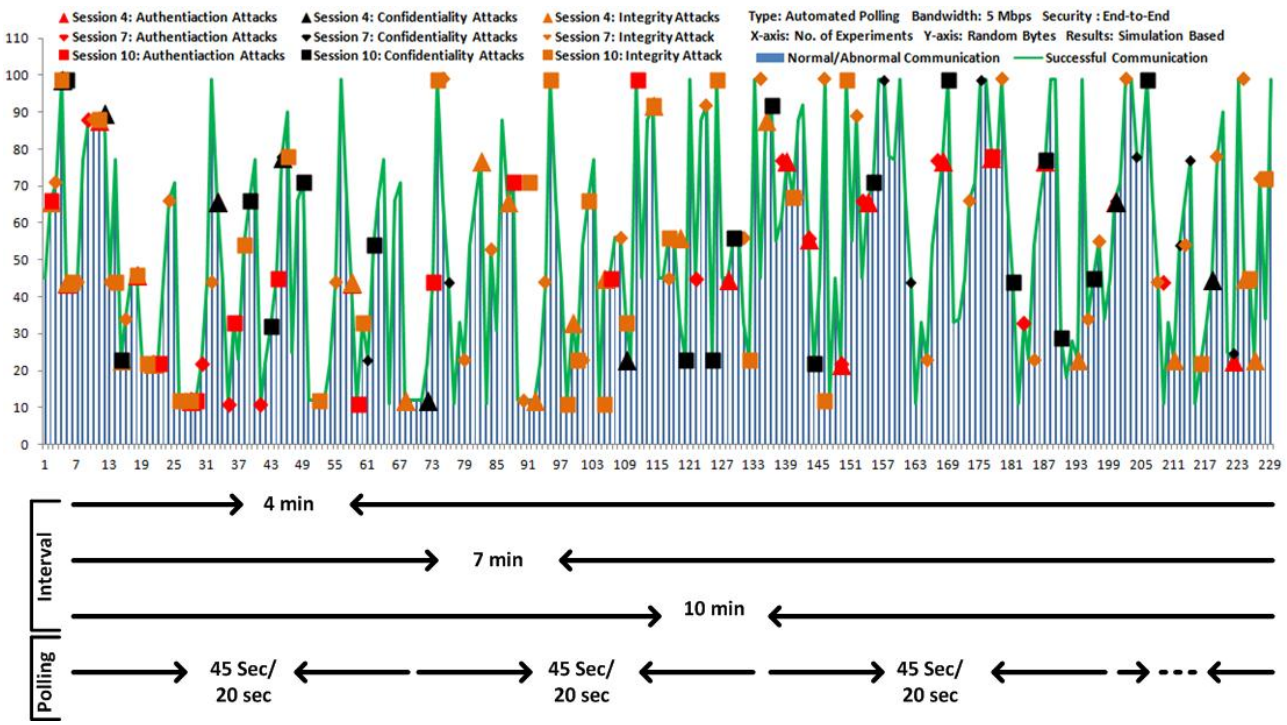


Figure 14. Automated polling (Scenario 3): Attack detection at specified intervals.

In each scenario, 229 experiments were individually performed at intervals of four, seven and ten minutes, to compute the level of attack detection. From the total number of experiments, one experiment was optionally employed to test the non-repudiation function. Therefore, a total of 228 experiments were performed at each interval during the calculation of attack detection percentage(s). The formula and other experimental details that are used to compute the attack detection percentage are as follows:

Attack Detection (%) = $\sum_{k=0}^{n}\left(Comp(x^{k_i}), Comp(y^{k_j}), Comp(z^{k_l})\right)/\sum_{e=1}^{n} a^{e} \times 100$

The values $'x, y, z'$ compute the attacks which are detected at specified intervals, and added to compute the total. The value $'a'$ computes and adds the number of experiments performed at each interval. Where, $'k'$ and $'e'$ show the sequence with limit $'n'$.

For example, experiments per interval and total experiments per scenario are: 228 and 684.

As the performance results show in Figure 15, total attack detection percentages of performance Figures 12–14 are calculated, and corresponding securities are measured. In the first scenario, the total attack detection percentage is computed as 95% and the corresponding security is 5%. These results show the lack of security design in DNP3. On the other side, in the second and third scenarios, total attack detection percentages are computed as 3% and 24% and corresponding computed securities are 97% and 76%. The security percentage is computed from the second scenario and this shows the significant enhancements of DNP3 security in automated polling during open connectivity with protocols such as TCP/IP. This also shows great enhancements compared to the first scenario and third scenario and with existing end-to-end developments [8–14,16–19,22,24,34,50,52].

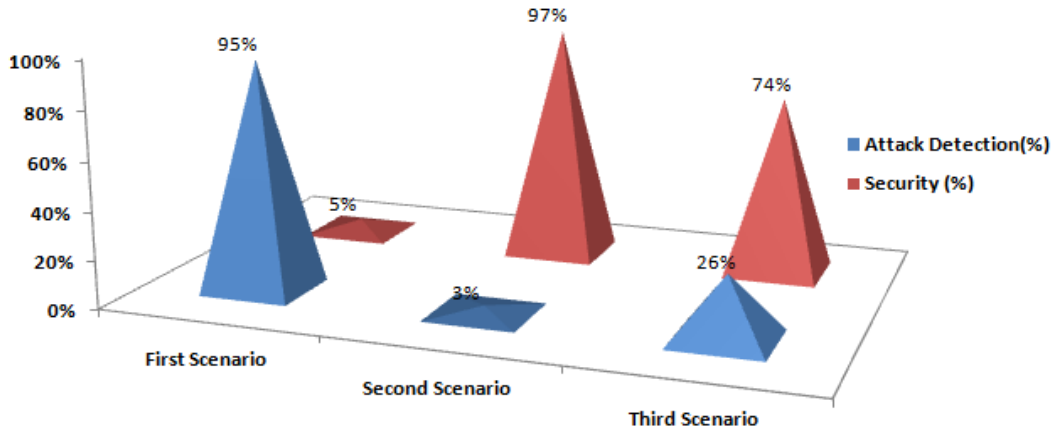


Figure 15. Performance comparison.

9. Related Work

A simulation has been designed, and two modules are proposed for SCADA/DNP3 system security enhancement [16,17]. The first module is tested without any security concerns in order to measure the level of SCADA security and compare it with other security enhancement modules. The second module is employed which uses a digital signature technique during the transmission of bytes between SCADA nodes. A security method called authentication octets is appended with bytes that are transmitted from the master node to the remote node in the SCADA testbed [16,17,21]. Upon receiving a message, the remote node uses a private key to decrypt the message. It then calculates and compares a hash value with the send hash digest. The third module was designated as a challenge response with random key sessions, which protect communication against attacks including data replay and modification and spoofing [8,16,17,51,53]. As analyzed, the security approach is not convenient for SCADA multicasting and broadcasting communications, because asymmetric encryption takes much time during processing and number of keys are also acquired during bytes' security. However, the above security approaches are relevant for SCADA unicasting or one-to-one communication [8,23,54,55].

The SCADA platform is more vulnerable than traditional or informative networks due to its connectivity with proprietary and non-proprietary protocols. The existing SCADA systems were not designed with cyber security in mind, but today these systems are interlinked with corporate networks as well as open protocols and ready to access information/data from distant places via the Internet [8,56,57]. While connected with open IP networks, SCADA systems are facing a number of challenges in the absence of a security mechanism that provides protection against cyber-attacks. Several security solutions have been employed to enhance the level of SCADA security but these solutions also have disadvantages while providing protection against Internet attacks [9,37,52,57–66]. Therefore, a security solution that provides protection while SCADA interacts with advanced networks or/and protocols (such as LAN/WAN) should be acquired [6,52,59–62,67–69].

10. Conclusions and Future Work

SCADA systems are gaining popularity in modern technology day-by-day; these systems are in most demand in industrial processing. However, such advances in technology development create vulnerabilities, which are dangerous for SCADA communication platforms. Thus, a cryptography solution was implemented within DNP3 protocol as part of a SCADA system during automated polling. The design and computed measurements act as symbols to examine and prove the validity of the proposed development, but the scope is limited to simulation. In this study, the word "critical" has been used to designate the automated polling because, in a polling scenario, an attacker has several chances to worm the SCADA traffic. This study outlines new research trends for SCADA automated polling and its security enhancements.

In future work, certificate authority (CA) is required, and the cryptography based security developments will be analyzed and, subsequently, cryptography algorithms will be selected as a potential solution for SCADA broadcasting/multicasting communication vulnerabilities where a number of nodes are configured and connected to the main controller(s). The network nodes would also be followed by both SCADA/DNP3 bounded and unbounded communication methods.

Acknowledgments

This paper was supported by the research funds of Wonkwang University in 2015.

Author Contributions

In this research, Aamir Shahzad and Malrey Lee conceived and designed the experiments; Aamir Shahzad and Hyung Doo Kim performed the experiments; Aamir Shahzad and Seon-mi Woo analyzed the data; Aamir Shahzad, Malrey Lee and Naixue Xiong contributed materials/analysis tools; Aamir Shahzad and Malrey Lee wrote the paper.

Conflicts of Interest

The authors declare no conflict of interest.

References

1. 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, 2006; pp. 2–13.
2. 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
3. Clarke, G.; Reynders, D.; Wright, E. *Practical Modern SCADA Protocols*; DNP3, 60870.5 and Related Systems; Elsevier: New York, NY, USA, 2004; pp. 73–129.
4. 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.
5. DNP Users Group. *DNP3 Application Layer Specification, Version 2.00*; DNP Organization: Washington, WA, USA, 2005; Volume 2.
6. Gao, J.; Liu, J.; Rajan, B.; Nori, R. SCADA Communication and Security Issues. *Secur. Commun. Netw.* **2014**, *7*, 175–194.
7. 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.
8. Musa, S.; Shahzad, A.; Aborujilah, A. Secure Security Model Implementation for Security Services and Related Attacks Based on End-to-End, Application Layer and Data Link Layer Security. In Proceedings of the 7th International Conference on Ubiquitous Information Management and Communication, Kota Kinabalu, Malaysia, 17–19 January 2013.
9. 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.
10. Hieb, J.L.; Graham, J.H.; Patel, S.C. *Cyber Security Enhancements for SCADA and DCS Systems*; Intelligent Systems Research Laboratory; Technical Report ISRL-TR-07-02; University of Louisville: Louisville, KY, USA, 2007.
11. 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, doi:10.1007/978-0-387-75462-8_10.
12. Kang, D.-J.; Kim, H.-M. 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.
13. 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**, *7*, 1670–1690.
14. 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.
15. Ali, M.; Khan, S.U.; Vasilakos, A.V. Security in Cloud Computing: Opportunities and Challenges. *Inf. Sci.* **2015**, *305*, 357–383.

16. 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. Commun. Netw.* **2014**, 1210–1218.
17. Lim, S.; Lee, E.; Park, C.-M. Equivalent Public Keys and a Key Substitution Attack on the Schemes from Vector Decomposition. *Secur. Commun. Netw.* **2014**, 1274–1282.
18. Patel, S.C. Secure Internet-Based Communication Protocol for SCADA Networks. Ph.D. Thesis, University of Louisville, Louisville, KY, USA, 2006.
19. Patel, S.C.; Bhatt, G.D.; Graham, J.H. Improving the Cyber Security of SCADA Communication Networks. *Commun. ACM* **2009**, *52*, 139–142.
20. Igure, V.M.; Laughter, S.A.; Williams, R.D. Security Issues in SCADA Networks. *Comput. Secur.* **2006**, *25*, 498–506.
21. Elsaid, W.H. Enhanced Cryptographic Approaches for SCADA Network Security. Ph.D. Thesis, University of Louisville, Louisville, KY, USA, 2010.
22. Saxena, A.; Pal, O.; Saquib, Z. Public Key Cryptography Based Approach for Securing SCADA Communications. In *Computer Networks and Information Technologies*; Communications in Computer and Information Science; Springer Berlin Heidelberg: Berlin, Germany, 2011; Volume 142, pp. 56–62.
23. Shahzad, A.; Musa, S.; Irfan, M. *N*-Secure Cryptography Solution for SCADA Security Enhancement. *Trends Appl. Sci. Res.* **2014**, *9*, 381–395.
24. Drahansky, M.; Balitanas, M. Cipher for Internet-based Supervisory Control and Data Acquisition Architecture. *J. Secur. Eng.* **2011**, *8*, 337–348.
25. Shbib, R.; Zhou, S.; Alkadhimi, K. SCADA System Security, Complexity, and Security Proof. In Proceedings of the ICPCA-SWS 2012, LNCS 7719, Istanbul, Turkey, 28–30 November 2012; pp. 405–410.
26. Ryu, D.H.; Kim, H.; Um, K. Reducing Security Vulnerabilities for Critical Infrastructure. *J. Loss Prev. Process Ind.* **2009**, *22*, 1020–1024.
27. Shyamasundar, R.K. Security and Protection of SCADA: A Big Data Algorithmic Approach. In Proceedings of the 6th International Conference on Security of Information and Networks (SIN'13), Aksaray, Turkey, 26–28 November 2013; ACM: New York, NY, USA, 2013; pp. 20–27.
28. Cardenas, A.A.; Amin, S.; Lin, Z.-S.; Huang, Y.-L.; Huang, C.-Y.; Sastry, S. Attacks against Process Control Systems: Risk Assessment, Detection, and Response. In Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security (ASIACCS'11), Hongkong, China, 22–24 March 2011; ACM: New York, NY, USA, 2011; pp. 355–366.
29. Ralston, P.A.S.; Graham, J.H.; Hieb, J.L. Cyber Security Risk Assessment for SCADA and DCS networks. *ISA Trans.* **2007**, *46*, 583–594.
30. Gold, S. The SCADA Challenge: Securing Critical Infrastructure. *Netw. Secur.* **2009**, *2009*, 18–20.
31. DNP Users Group. *DNP3 Specification, Secure Authentication*; DNP Organization: Washington, WA, USA, 2010; Supplement to Volume 2.
32. 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.

33. East, S.; Butts, J.; Papa, M.; Shenoi, S. A Taxonomy of Attacks on the DNP3 Protocol. In *Critical Infrastructure Protection III*; Springer Berlin Heidelberg: Berlin, Germany, 2009; pp. 67–81.
34. Mander, T.; Cheung, R.; Nabhani, F. Power System DNP3 Data Object Security Using Data Sets. *Comput. Secur.* **2010**, *29*, 487–500.
35. Shahzad, A.; Kalum, P.U.; Young, K.L.; Soojin, P.; Malrey, L. The Sensors Connectivity within SCADA Automation Environment and New Trends for Security Development during Multicasting Routing Transmission. *Int. J. Distrib. Sens. Netw*. **2015**, in press.
36. Lee, D.; Kim, H.; Kim, K.; Yoo, P.D. Simulated Attack on DNP3 Protocol in SCADA System. In Proceedings of the The 31th Symposium on Cryptography and Information Security, Kagoshima, Japan, 21–24 January 2014.
37. Mohammadi, N.B.; Misic, J.; Misic, V.B.; Khazaei, H. A Framework for Intrusion Detection System in Advanced Metering Infrastructure. *Secur. Commun. Netw.* **2014**, *7*, 195–205.
38. Mirkovic, J.; Reiher, P. A Taxonomy of DDoS Attack and DDoS Defense Mechanisms. *ACM SIGCOMM Comput. Commun. Rev.* **2004**, *34*, 39–53.
39. Jin, D.; Nicol, D.M.; Yan, G. An Event Buffer Flooding Attack in DNP3 Controlled SCADA Systems. In Proceedings of the 2011 Winter Simulation Conference (WSC), Phoenix, AZ, USA, 11–14 December 2011; pp. 2614–2626.
40. Shahzad, A.; Xiong, N.; Irfan, M.; Lee, M.; Hussain, S.; Khaltar, B. A SCADA intermediate simulation platform to enhance the system security. In Proceedings of the 2015 17th International Conference on Advanced Communication Technology (ICACT), Seoul, Korea, 1–3 July 2015; pp. 368–373.
41. Graham, J.; Patel, S. *Security Considerations in SCADA Communication Protocols*; Technical Report TR-ISRL-04-01; Intelligent Systems Research Laboratory: Louisville, KY, USA, 2004
42. Rrushi, D.; di Milano, U. SCADA Intrusion Prevention System. In Proceedings of the 1st CI2RCO Critical Information Infrastructure Protection Conference, Hampshire, UK, 14 March 2006.
43. Bompard, E.; Gao, C.; Napoli, R.; Russo, A.; Masera, M.; Stefanini, A. Risk Assessment of Malicious Attacks Against Power Systems. *IEEE Trans. Syst. Man Cybern. A Syst. Hum.* **2009**, *39*, 1074–1085.
44. Fernandez, J.; Fernandez, A. SCADA Systems: Vulnerabilities and Remediation. *J. Comput. Sci. Coll.* **2005**, *20*, 160–168.
45. Patel, S.; Yu, Y. Analysis of SCADA Security models. *Int. Manag. Rev.* **2007**, *3*, 68–76.
46. Faruk, A. Testing & Exploring Vulnerabilities of the Applications Implementing DNP3 Protocol. Masters' Dissertation. Royal Institute of Technology, Stockholm, Sweden, 2008.
47. Hong, S.; Lee, S. Challenges and Perspectives in Security Measures for the SCADA System. In Proceedings of the 5th Myongji-Tsinghua University Joint Seminar on Protection & Automation, Korea, 2008.
48. Fujisaki, E.; Okamoto, T. Secure Integration of Asymmetric and Symmetric Metric Encryption Schemes. In *Advances in Cryptology—CRYPTO'99*; LNCS; Springer Berlin Heidelberg: Berlin, Germany, 1999; Volume 1666, pp. 537–554.
49. Rivest, R.L.; Shamir, A.; Adleman, L. A method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Commun. ACM* **1978**, *21*, 120–126.
50. 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.

51. Shahzad, A.; Musa, S.; Irfan, M.; Asadullah, S. Deployment of New Dynamic Cryptography Buffer for SCADA Security Enhancement. *J. Appl. Sci.* **2014**, *14*, 2487–2497.
52. Liyanage, M.; Gurtov, A. Securing Virtual Private LAN Service by Efficient Key Management. *Secur. Commun. Netw.* **2014**, *7*, 1–13.
53. Chandia, R.; Gonzalez, J.; Kilpatrick, T.; Papa, M.; Shenoi, S. Security Strategies for SCADA Networks. *IFIP Int. Fed. Inf. Process.* **2008**, *253*, 117–131.
54. Rong, C.; Nguyen, S.T.; Jaatun, M.G. Beyond Lightning: A Survey on Security Challenges in Cloud Computing, Special Issue on Recent Advanced Technologies and Theories for Grid and Cloud Computing and Bio-Engineering. *Comput. Electr. Eng.* **2013**, *39*, 47–54.
55. Riaz, R.; Naureen, A.; Akram, A.; Akbar, V.; Kim, K.H.; Farooq Ahmed, H. A Unified Security Framework with Three Key Management Schemes for Wireless Sensor Networks. *Comput. Commun.* **2008**, *31*, 4269–4280.
56. Mohamed, F.A.; Hemayed, E.E. Using Trusted Computing in Trusted Mail Transfer Protocol. *Secur. Commun. Netw.* **2014**, *7*, 926–933.
57. 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.
58. 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.
59. 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.
60. Morris, T.H.; Gao, W. Industrial control system cyber attacks. In Proceedings of the 1st International Symposium on ICS & SCADA Cyber Security Research, 2013; BCS: UK; pp. 22–29.
61. Robles, R.J.; Balitanas, M.; Kim, T. Security Encryption Schemes for Internet SCADA: Comparison of the Solutions. *Commun. Comput. Inf. Sci.* **2011**, *223*, 19–27.
62. Loutchkina, I.; Jain, L.C.; Nguyen, T.; Nesterov, S. Systems' Integration Technical Risks' Assessment Model (SITRAM). *IEEE Trans. Syst. Man Cybern. Syst.* **2014**, *44*, 342–352.
63. Wang, L.; Ren, S.; Korel, B.; Kwiat, K.A.; Salerno, E. Improving System Reliability against Rational Attacks Under Given Resources. *IEEE Trans. Syst. Man Cybern. Syst.* **2014**, *44*, 446–456.
64. Lin, C.-H.; Song, K.-T. Probability-Based Location Aware Design and on-Demand Robotic Intrusion Detection System. *IEEE Trans. Syst. Man Cybern. Syst.* **2014**, *44*, 705–715.
65. Jiang, Y.; Jiang, J.C. Diffusion in Social Networks: A Multiagent Perspective. *IEEE Trans. Syst. Man Cybern. Syst.* **2015**, *45*, 198–213.
66. Ko, J.; Lim, H.; Lee, S.; Shon, T. AVQS: Attack Route-Based Vulnerability Quantification Scheme for Smart Grid. *Sci. World J.* **2014**, *2014*, 1–6.
67. Robles, R.-J.; Balitanas, M.; Caytiles, R.; Gelogo, Y.; Kim, T. Comparison of Encryption Schemes as Used in Communication between SCADA Components. In Proceedings of the 2011 International Conference on Ubiquitous Computing and Multimedia Applications (UCMA), Daejeon, Korea, 13–15 April 2011; pp. 115–118.

68. Scacchioli, A.; Rizzoni, G.; Salman, M.A.; Li, W.; Onori, S.; Zhang, X. Model-based Diagnosis of an Automotive Electric Power Generation and Storage System. *IEEE Trans. Syst. Man Cybern. Syst.* **2014**, *44*, 72–85.
69. Eirinaki, M.; Louta, M.D.; Varlamis, I. A Trust-Aware System for Personalized User Recommendations in Social Networks. *IEEE Trans. Syst. Man Cybern. Syst.* **2014**, *44*, 409–421.