Journal of Applied Sciences

ISSN 1812-5654

science alert

ANSInet
an open access publisher
http://ansinet.com

Journal of Applied Sciences 14 (20): 2487-2497, 2014
ISSN 1812-5654 / DOI: 10.3923/jas.2014.2487.2497

Deployment of New Dynamic Cryptography Buffer for SCADA Security Enhancement

[1]A. Shahzad, [1]S. Musa, [2]M. Irfan and [3]S. Asadullah
[1]Malaysian Institute of Information Technology (MIIT), 1016,
Jalan Sultan Ismail, Universiti Kuala Lumpur, 50250, Kuala Lumpur, Malaysia
[2]Windfield College, Paser Seni, Kuala Lumpur, Malaysia
[3]Kulliyah of Information and Communication Technology, International Islamic University, Malaysia

Abstract: The current study is based on novel solution which deploy the security mechanism, more advance cryptography solution within Distributed Network Protocol (DNP3) stack as a part of critical system (or SCADA system). The "Dynamic Cryptography Buffer (DCB)" has been implemented that contains 56 bytes from total size of "Application Protocol Data Unit (APDU) bytes" as a part of application layer of DNP3 protocol. The DCB contains several fields/subfields which have been used during implementation of cryptography algorithms and other information (or detail) related with protocol security. During implementation within DNP3 protocol, the bytes are dynamically stored after processing (security deployment) within DCB, without affecting the total size of DNP3 protocol stack. This novel study gives new directions for SCADA or its protocols security deployment and enhancement.

Key words: DNP3 protocol stack, DNP3 protocol security, dynamic buffer, dynamic bytes, cryptography algorithms, performance results

INTRODUCTION

The SCADA system has been used several protocols "such as profibus protocol, modbus protocol, DNP3 protocol, foundation fieldbus protocol, IEC 60870-5 protocol and other series, modbus plus protocol and data highway plus/DH-485 protocol", for the purpose of industrial automation and processing. Each protocol has distinct specifications during installation, configuration and transmission services. The Distributed Network Protocol (DNP3) is most important and famous protocol that has been used in industrial processing (or SCADA processing) (Shahzad and Musa, 2012; Shahzad *et al*., 2013).

The DNP3 protocol has contains four layers included "application layer, pseudo-transport layer, data link layer and physical layer", in its stack and used "Transport Control Protocol/Internet Protocol (TCP/IP)" over internet. The application layer of DNP3 protocol takes random bytes from upper layer (user application layer) and able to process 2048 bytes to lower layer. The each size of Transport Protocol Data Unit (TPDU) is up to 250 bytes with header byte while data link layer is able to receive 250 bytes from pseudo-transport layer and transmit 292 bytes with 32 Cyclic Redundancy Checker (CRC) bytes to physical layer included 10 bytes of header (Musa *et al*., 2013a, b; Shahzad *et al*., 2014a).

NEW DNP3 PROTOCOL SECURITY STACK

The new DNP3 protocol security stack has number of fields/subfields which are utilized during cryptography implementation (proposed implementation). The detail related with fields/subfields is as followed.

Source address and port: This is 2 bytes (unassigned) field within stack which identifies the source address and port, either master station send request or remote station send response.

Destination address and port: This is also 2 bytes (unassigned) field within stack which identifies the destination address and port, either master station received response message or remote station received request message.

User bytes: This field defines the bytes (user bytes) that are being used by security (proposed solution) implementation within each layer.

Corresponding Author: A. Shahzad, Malaysian Institute of Information Technology (MIIT), 1016, Jalan Sultan Ismail, Universiti Kuala Lumpur, 50250, Kuala Lumpur, Malaysia

- **User bytes:** Application Protocol Data Unit (APDU) size is up to 1992 bytes and Application Protocol Control Information (APCI) is 2 or 4 bytes within APDU, case of request or response
- **User bytes:** Transport Protocol Data Unit (TPDU) size is up to 250 bytes and Transport Protocol Control Information (TPCI) has 1 bytes within TPDU, both request and response message (bytes)
- **User bytes:** Link Protocol Data Unit (LPDU) size is up to 292 bytes (with optional CRC bytes) and Link Protocol Control Information (LPCI) has 10 bytes within LPDU, both for request and response

Cryptography key sequence: This is 4 bytes (unassigned) subfield within stack (cryptography implementation field) and used to keep the track of cryptography keys during message (bytes) generation within each layer of DNP3 protocol. Each time security is implemented in layers include application layer, pseudo-transport layer and data link layer (Musa *et al*., 2013a; Shahzad *et al*., 2014b), cryptography counter is updated and value is added in cryptography key sequence field. The two flags are used within each layer to monitor the status of security (cryptography implementation). More detail is depicted in Table 1.

This field also keeps the record of message (bytes) during communication. Each time message (bytes) is transmitted to remote station, counter is incremented by one and value is update in cryptography key sequence field.

Dynamic storage (bytes): This is dynamic byte subfield within stack (cryptography implementation field) and used to store cryptography implementation information. In start, this subfield occupy 16-56 bytes within stack, either message (bytes) request or response. In case, dynamic buffer is full and acquired additional space for cryptography implementation than CRC (32 bytes) from data link layer are used for additional bytes storage.

Optional (bytes): This is 2 bytes (unassigned) subfield within stack (cryptography implementation field) and usually implemented at the end of DNP3 stack or when message is ready to transmit (request or response). This subfield is used to verify all contents during message (bytes) generation process.

Padding bytes: This subfield occupies dynamic bytes within stack (cryptography implementation field). In case, dynamic buffer has extra memory space than this space is filled with padding bytes. This function indicates that message (byte) has been constructed and ready for transmit (send or response) to desire station.

Table 1: Cryptography key sequence status

Flages	Status
APDU: Flag (0)	APDU bytes: Security_fail or status unknown
APDU: Flag (1)	APDU bytes: Security_successful or status successful
TPDU: Flag (0)	TPDU bytes: Security_fail or status unknown
TPDU: Flag (1)	TPDU bytes: Security_successful or status successful
LPDU: Flag (0)	LPDU bytes: Security_fail or status unknown
LPDU: Flag (1)	LPDU bytes: Security_successful or status successful

Acknowledgment: This is 2 bytes (unassigned) subfield within stack (cryptography implementation field). During communication, master station send request message (bytes) with acknowledgment flag (set) than upon receiving, remote station also send acknowledge message to mater station, not data.

- ACK, flag (0): Indicate master/remote station communication without acknowledgment
- ACK, flag (1): Indicate master/remote station communication with acknowledgment

Critical (bytes): This is optional 1 bytes (unassigned) subfield within stack (cryptography implementation field). In case, attacks are successfully within communication, this field shows the status or system behavior.

Non- critical (bytes): This is optional 1bytes (unassigned) subfield within stack (cryptography implementation field). In case, attacker attack but not successfully within communication, this field shows the status or system behavior.

Solution (select method): This is one bytes (unassigned) subfield within stack (cryptography implementation field) and used to identified which cryptography method is currently using for SCADA/DNP3 security (depend on requirement). Figure 1 illustrates the new DNP3 protocol security stack while Table 2 shows the detail fields/subfields description of new DNP3 protocol security stack.

RESULTS AND DISCUSSION

Dynamic buffer utilization and command request (bytes): The below APDU bytes (MTU request), logical 98 bytes (request dummy bytes) have been transmitted from MTU (sender) application layer to RTU (receiver) application layer. The below bytes (MTU application layer) are the request message being transmitted from MTU to RTU while the heighted bytes show the Application layer Header or AH information, object header information and security information. The other heighted byte, 0x00c3 is application control information and 0x0001 is the function code (uses for read request).

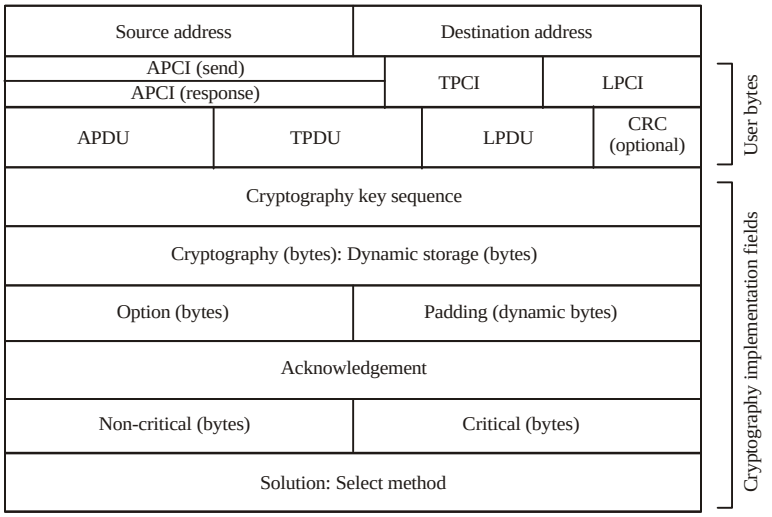

Fig. 1: New DNP3 protocol security stack

Table 2: Fields/subfields description: New DNP3 protocol security stack

Protocol (stack) fields	Protocol (stack) sub fields	Total bytes	Description
Source address and port		2 bytes (unassigned)	Source/destination fields
Destination address and port		2 bytes (unassigned)	
APDU	Request APCI: 2 bytes Response APCI: 4 bytes	1992 bytes	User fields (bytes)
TPDU	Request/response TPCI: 1 byte (each) Request/response LPCI: 10 bytes (each) CRC (optional): 32 bytes	250 bytes 292 bytes	
Dynamic cryptography buffer	Cryptography key sequence: 4 bytes (unassigned) Dynamic storage: 16-56 bytes Optional: 2 bytes (unassigned) Padding: 2 bytes (unassigned) Acknowledgement: 2 bytes (unassigned) Critical: 1 bytes (unassigned) Non-critical: 1 bytes (unassigned) Solution: select method: 1 bytes (unassigned)	Original size: 56 bytes included source and destination bytes During exception: Original bytes added with CRC bytes	Cryptography implementation fields

Request APDU bytes (logical) are presented as follows:

```
ae 2b 8d 6c 8d 6c af ca ee ee 98 0d ba cc 6e 6c cf
fa 84 ba ea fa 6e a2 b8 1e cf 0d ba ea fa a7 ba cf
fa ca ee cc a2 aa 4c 1e cf 0d 2b 8d 6c af 1e 8d 34
4c 2b ba ea fa 4c fe 1e cf 0d ef fa 2b ba 6e a7 ba
0d cc ba ea fa a7 6e ec 2b ba ea fa aa ca bf ca cf
c3 01 ie 02 00 04 07 1e ee 1a ee 2a ee
```

The bytes, 0x00ie and 0x0002 are designated for group and variation fields. The byte, 0x0000 is qualifier field which is used both in request/response message while data object are acquired (needed) from contiguous indexes. The bytes, 0x0004-0x0007 are designated for range fields. The remaining bytes 0x001e, 0x000ee, 0x001a, 0x00ee, 0x0002a and 0x00ee are the cryptography information (bytes) which have been utilized for security implementation.

The bytes such as:

$$OS_{0090}.R_{0008}.C_{00XX}.B_{00XX}\_\_OS_{0090}.R_{0008}.C_{00XX}.B_{00XX}$$
(APDU bytes)

$$OS_{0190}.R_{0008}.C_{00XX}.B_{00XX}\_\_OS_{0190}.R_{0008}.C_{00XX}.B_{00XX}$$
(TPDU bytes)

and:

$$OS_{0290}.R_{0008}.C_{00XX}.B_{00XX}\_\_OS_{0290}.R_{0008}.C_{00XX}.B_{00XX}$$
(LPDU bytes)

with offsets: 0x0090, 0x0190 and 0x0290 are designated for dynamic buffer implementation while bytes:

$$OS_{10}.R_{0009}.C_{00XX}.B_{0000}\_\_OS_{0100}.R_{0009}.C_{00XX}.B_{00XX}$$
(APDU bytes)

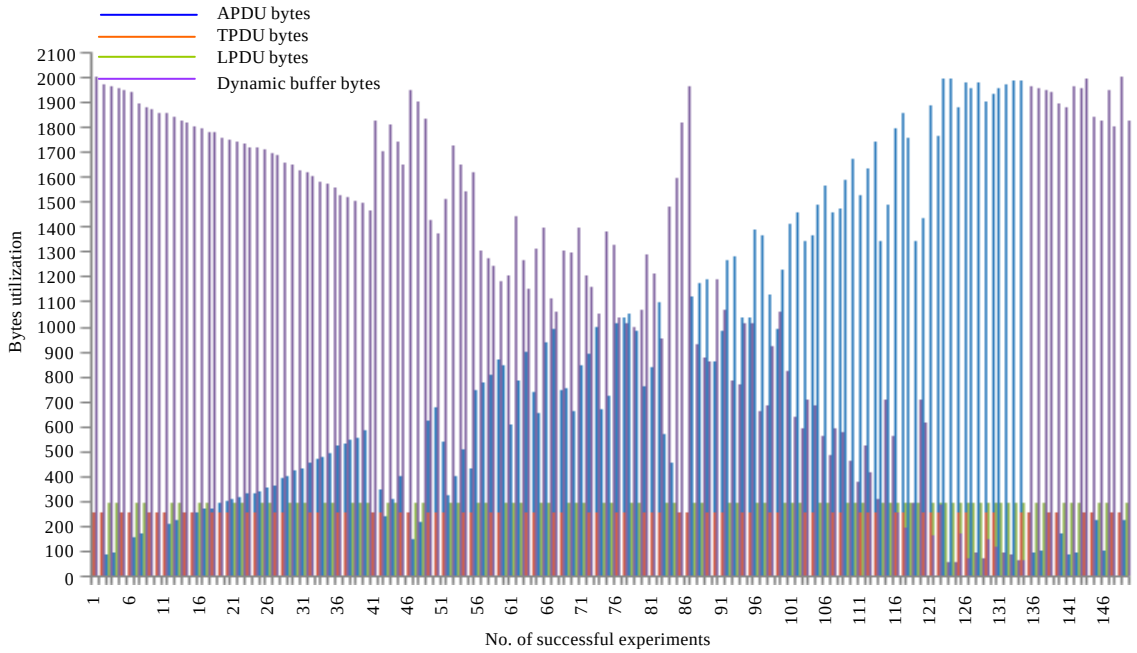

Fig. 2: Bytes allocation for dynamic buffer during communication

$$OS_{0200}.R_{0009}.C_{00XX}.B_{0000}\_\_OS_{0200}.R_{0009}.C_{00XX}.B_{00XX}$$
(TPDU bytes)

and:

$$OS_{0300}.R_{0009}.C_{00XX}.B_{0000}\_\_OS_{0300}.R_{0009}.C_{00XX}.B_{00XX}$$
(LPUD bytes)

are reserved for future (implementation). Where, 'XX 'are the numbers of bytes utilized (added), upon the requirements of dynamic buffer implementation.

At the time of initialization (MTU request/RTU response), the size of dynamic buffer is 56 bytes and application layer (APDU) buffer is able to store 1992 bytes in both request and response message. If APDU size is less than 1992 bytes, than the remaining bytes will dynamically allocate for dynamic buffer (used to store information related with security implementation). If the dynamic buffer is full, mean that cryptography information has been stored successfully and still memory (space) is remaining within buffer. Then remaining space will fill with padding bytes, to create full packed APDU fragment with security (cryptography solution) implementation. Figure 2, shows the number of bytes utilization process within proposed DNP3 security stack with dynamic buffer bytes.

In Fig. 3, dynamic buffer size (bytes) has been increased as decreasing of APDU size (bytes) within DNP3 stack while Transport Protocol Unit (TPDU) and link protocol unit (LPDU) bytes are stable (constant) as 250 bytes and 292 bytes or assume that TPDU and LPDU bytes are fixed for each process (testbed experiment).

In Fig. 3, red markers show that 8 times dynamic buffer has been full and acquired extra memory space for cryptography implementation (storage). In this case, exception message has been executed and than 32 bytes from CRC (data link layer, user data blocks only) are shifted (move) to dynamic buffer (memory space). Thus, these 32 bytes are further utilized for cryptography deployment within DNP3 protocol. Figure 4, shows the dynamic buffer status while usage of extra 32 bytes from CRC (data link layer, user data blocks only).

RESULTS AND DISCUSSION

Dynamic buffer utilization and command response (bytes): The below bytes (RTU response), logical 107 bytes (response dummy bytes) have been reply back logically from RTU (sender or response) application layer to MTU (receiver) application layer. The below bytes (RTU application layer), are the response (bytes) transmitted from RTU to MTU while the heighted bytes show the application layer header or AH information, object header information and security information (proposed solution).

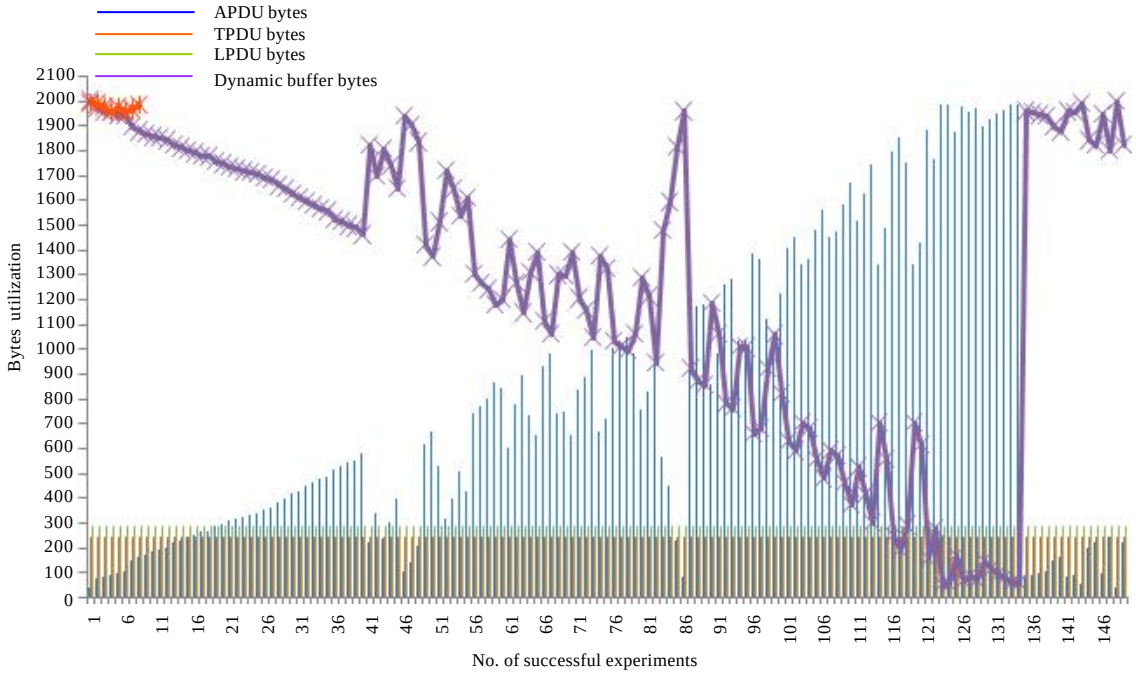

Fig. 3: Dynamic buffer bytes status (with exception) during communication

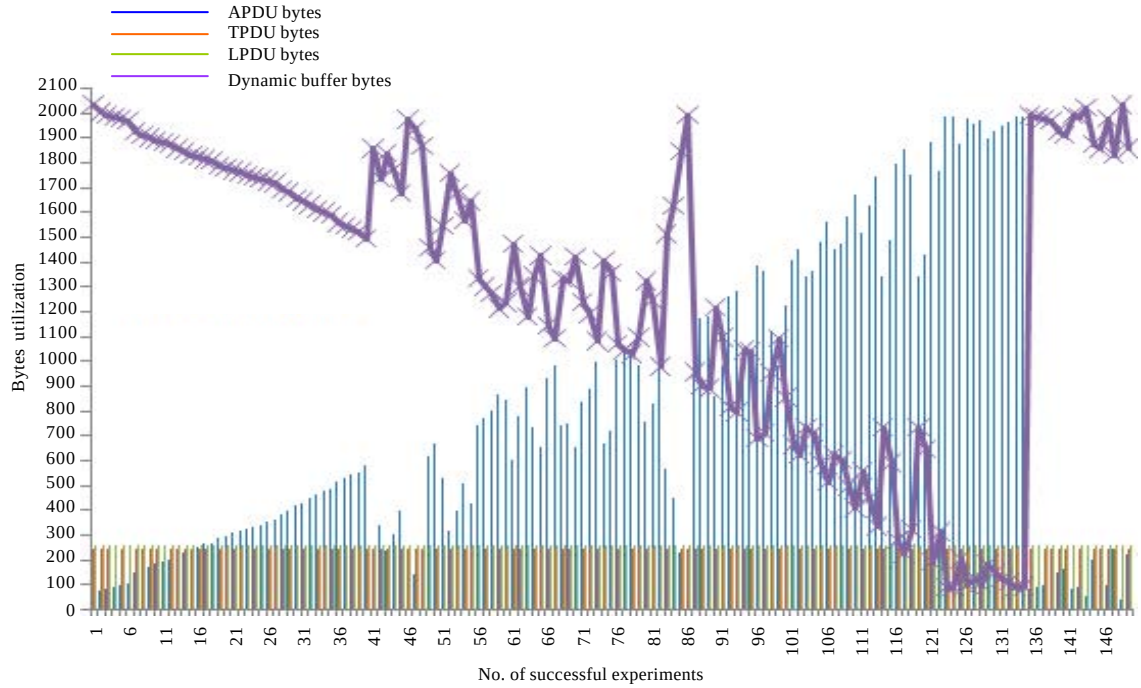

Fig. 4: Dynamic buffer bytes status (without exception) during communication

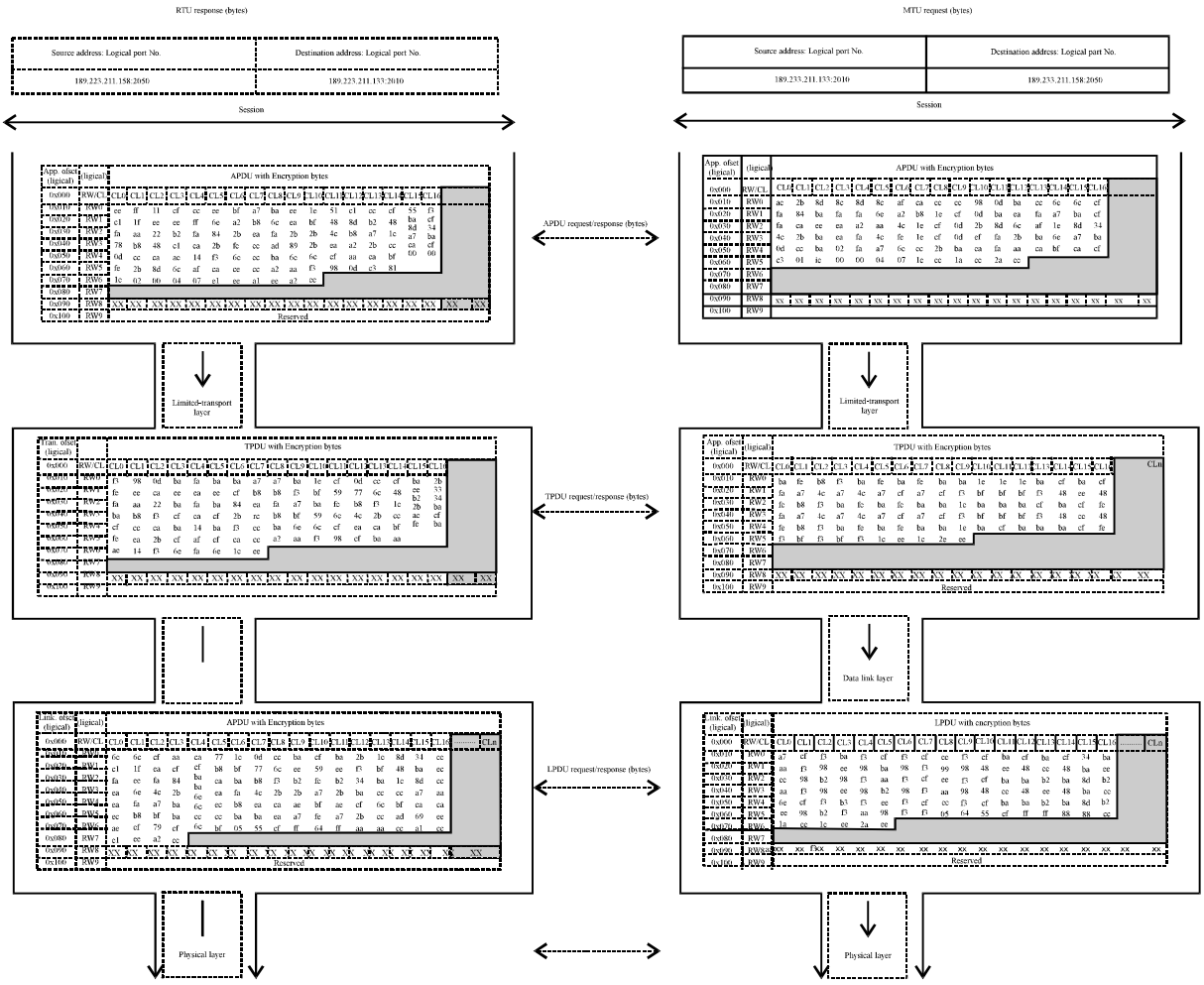

Fig. 5: Communication between DNP3 protocol stack (request/response bytes)

Response APDU bytes (logical) are presented as follows:

```
ee ff aa cf cc ee bf a7 ba ee 1e 51 c1 cc cf 55 f3
c1 1f ee ee ff 6e a2 b8 6c ea bf 48 8d b2 48 ba cf
fa aa 22 b2 fa 84 2b ea fa 2b 2b 4c b8 a7 1e 8d 34
78 b8 48 c1 ea 2b fe ee ad 69 2b ea a2 2b cc a7 ba
0d cc ca ae 14 f3 6e ec ba 6e 6c cf aa ca bf ca cf
fe 2b 8d 6c af ca ee cc a2 aa f3 98 0d c3 81 00 00
1e 02 00 04 07 e1 ee a1 ee a2 ee
```

The heighted byte, 0x00c3 is application control information and 0x0081 is the function code (used for send response). Additional, two bytes 0x0000 and 0x0000 have been added within response message that distinguish the response message (bytes) header from request message (bytes) header. The internal indication or IIN (In response header fields) is two bytes fields follow by function code and used to send response message to master station. Each bit within IIN fields ($11N_{S(1.0)}$__$11N_{E(1.7)}$ and $11N_{S(2.0)}$__$11N_{E(2.7)}$) has specific meaning of sending response message.

The dynamic buffer implementation and bytes utilization process within response message (bytes) is identical as request message (bytes). Mean that whole dynamic buffer process is same within request and response message (bytes).

The bytes, 0x00ie and 0x0002 are designated for group and variation fields. The byte, 0x0000 is qualifier field which is used both in request/response message while data object are acquired (needed) from contiguous indexes. The bytes, 0x0004__0x0007 are designated for range fields. The reaming bytes 0x00e1, 0x000ee, 0x00a1, 0x00ee, 0x000a2 and 0x00ee are the cryptography information (bytes) which have been utilized for security implementation (proposed). Pseudo-transport layer and data link layer specifications of response bytes

Table 3: Transfer_functions bytes with cryptography implementation

MTU/RTU Communication	Application Control (AC)	Function Code (FC)	Internal Indication (IIN)		Object field		Qualifier	Range		Encryption bytes						Padding bytes (optional)
0H0000(hex)			IIN_{1x}	IIN_{2x}	Group	Variation		Start	Stop							
Read function																
MTU request	c3	01			1e	02	00	04	07	1e	ee	1a	ee	2a	ee	
RTU response	c3	81	00	00	1e	02	00	04	07	e1	ee	a1	ee	a2	ee	
Write function																
MTU request	c3	01			32	01	07	01		1e	ee	1a	ee	2a	ee	
RTU response	c3	81	00	00						e1	ee	a1	ee	a2	ee	

Table 4: Control_functions bytes with cryptography implementation

MTU/RTU Communication	Application Control (AC)	Function Code (FC)	Internal Indication (IIN)		Object field		Qualifier	Range		Encryption bytes						Padding bytes (optional)
0H0000(hex)			IIN_{1x}	IIN_{2x}	Group	Variation		Start	Stop							
Select function																
MTU request	c3	03			OC	01	17	01		1e	ee	1a	ee	2a	ee	
RTU response	c3	81	00	00	OC	01	17	01		e1	ee	a1	ee	a2	ee	
Operate function																
MTU request	c3	04			OC	01	17	01		1e	ee	1a	ee	2a	ee	
RTU response	c4	81	00	00	OC	01	17	01		e1	ee	a1	ee	a2	ee	
Direct operate function																
MTU request	c3	05			OC	01	17	01		1e	ee	1a	ee	2a	ee	
RTU response	c3	81	00	00	OC	01	17	01		e1	ee	a1	ee	a2	ee	
Direct operation function, no anknowledgment																
MTU request	c3	06			OC	01	17	01		1e	ee	1a	ee	2a	ee	
RTU response	c4	81	00	00	OC	01	17	01		e1	ee	a1	ee	a2	ee	

(message construction) have been remain same in both cases; in the case of MTU send request to RTU or RTU send response to MTU. Figure 5 illustrates the logical communication of DNP3 protocol stack (Layers) between request bytes and response bytes.

DNP3 application layer has specified numbers of function codes (dnp.org) followed by Application Control (AC) field that has been added within message (message header) during MTU/RTU or RTU/MTU communication. Function codes define the specific function for data (bytes) being operated or what function to be operating on data. Some functions are limited for specific data (bytes) on which, they are operates. Table 3-9 show the several functions deployment (utilization) in both cases; In the case of MTU send request to RTU and RTU send response to MTU (DNP3, 2005). Function codes are added within application layer header or AH without interaction of pseudo-transport layer and data link layer. The highlighted or colored cells in Table 3-9 show that the specified operations are not employed during SCADA/DNP3 transmition. The application layer functions deployment within message (request and response) has been distributed into following main parts:

- The function bytes (codes) 0x0000, 0x0001 and 0x0002, are the transfer_functions bytes (codes). These functions define data objects for transferring information (data) between master station and remote station (Read function) and/or remote station and master station (Write function). Table 3 show the deployment of transferring functions bytes (codes) in both MTU send request (message) and RTU send response (message) with security bytes (cryptography solution)
- The function bytes (codes) 0x0003, 0x0004, 0x0005 and 0x0006 are the control_functions bytes (codes).These functions are used to manage (operate) remote station control points (coming from field devices) depending on master station request. Table 4, show the deployment of control functions bytes(codes) in both

Table 5: Freeze_functions bytes with cryptography implementation

MTU/RTU Communication	Application Control (AC)	Function Code (FC)	Internal Indication (IIN)		Object field		Qualifier	Range		Encryption bytes						Padding bytes (optional)
0H0000(hex)			IIN_{1x}	IIN_{2x}	Group	Variation		Start	Stop							
Freeze function																
MTU request	c3	07			14	00	06			1e	ee	1a	ee	2a	ee	
RTU response	c3	81	00	00						e1	ee	a1	ee	a2	ee	
Freeze function, no acknowledgment																
MTU request	c3	08			14	00	06			1e	ee	1a	ee	2a	ee	
RTU response	c4	81	00	00						e1	ee	a1	ee	a2	ee	
Freeze and clear function																
MTU request	c3	09			14	00	06			1e	ee	1a	ee	2a	ee	
RTU response	c3	81	00	00						e1	ee	a1	ee	a2	ee	
Freeze and clear function, no acknowledgment																
MTU request	c3	0A			14	00	06			1e	ee	1a	ee	2a	ee	
RTU response	c4	81	00	00						e1	ee	a1	ee	a2	ee	
Freeze function-AT-Time																
MTU request	c3	0B			32	02	07	01		1e	ee	1a	ee	2a	ee	
RTU response	c3	81	00	00						e1	ee	a1	ee	a2	ee	
Freeze-Time-AT function, no aknowledgment																
MTU request	c3	0C			32	02	07	01		1e	ee	1a	ee	2a	ee	
RTU response	c4	81	00	00						e1	ee	a1	ee	a2	ee	

Table 6: Application_control_functions bytes with cryptography implementation

MTU/RTU Communication	Application Control (AC)	Function Code (FC)	Internal Indication (IIN)		Object field		Qualifier	Range		Encryption bytes						Padding bytes (optional)
0H0000(hex)			IIN_{1x}	IIN_{2x}	Group	Variation		Start	Stop							
Cold-restart function																
MTU request	c3	0D			34	00	07	01		1e	ee	1a	ee	2a	ee	
RTU response	c3	81	00	00	34	00	07	01		e1	ee	a1	ee	a2	ee	
Warm_restart function																
MTU request	c3	0E			34	00	07	01		1e	ee	1a	ee	2a	ee	
RTU response	c4	81	00	00	34	00	07	01		e1	ee	a1	ee	a2	ee	
Initialize_data function																
MTU request	c3	0F			32	02	07	01		1e	ee	1a	ee	2a	ee	
RTU response	c3	81	00	00						e1	ee	a1	ee	a2	ee	
Initialize_application function																
MTU request	c3	10			5a	01	5b	01		1e	ee	1a	ee	2a	ee	
RTU response	c3	81	00	00						e1	ee	a1	ee	a2	ee	
Start application function																
MTU request	c3	11			5a	01	5b	01		1e	ee	1a	ee	2a	ee	
RTU response	c3	81	00	00						e1	ee	a1	ee	a2	ee	
Stop application function																
MTU request	c3	12			5a	01	5b	01		1e	ee	1a	ee	2a	ee	
RTU response	c4	81	00	00						e1	ee	a1	ee	a2	ee	

MTU send request (message) and RTU send response (message) with security bytes (cryptography solution)

- The function bytes (codes) 0x0007, 0x0008, 0x0009, 0x000A, 0x000B and 0x000C are the freeze_functions bytes (codes). These functions are used to record

Table 7: Configuration_functions bytes with cryptography implementation

MTU/RTU Communication	Application Control (AC)	Function Code (FC)	Internal Indication (IIN)		Object field		Qualifier	Range		Encryption bytes						Padding bytes (optional)
0H0000(hex)			IIN_{1x}	IIN_{2x}	Group	Variation		Start	Stop							
Save configuration function																
MTU request	c3	13			5a	01	5b	01		1e	ee	1a	ee	2a	ee	
RTU response	c3	81	00	00						e1	ee	a1	ee	a2	ee	
Enable_unsolicited function																
MTU request	c3	14			32	01	07	01		1e	ee	1a	ee	2a	ee	
RTU response	c4	81	00	00						e1	ee	a1	ee	a2	ee	
Disable_unsolicited function																
MTU request	c3	15			3c	02	06			1e	ee	1a	ee	2a	ee	
RTU response	c3	81	00	00						e1	ee	a1	ee	a2	ee	
Assign class function																
MTU request	c3	16			3c	02	06			1e	ee	1a	ee	2a	ee	
RTU response	c4	81	00	00						e1	ee	a1	ee	a2	ee	

Table 8: Time_synchronization_functions bytes with cryptography implementation

MTU/RTU Communication	Application Control (AC)	Function Code (FC)	Internal Indication (IIN)		Object field		Qualifier	Range		Encryption bytes						Padding bytes (optional)
0H0000(hex)			IIN_{1x}	IIN_{2x}	Group	Variation		Start	Stop							
Delay measure function																
MTU request	c3	17								1e	ee	1a	ee	2a	ee	
RTU response	c3	81	00	00	34	02	07	01		e1	ee	a1	ee	a2	ee	
Record_current_time function																
MTU request	c3	18								1e	ee	1a	ee	2a	ee	
RTU response	c3	81	00	00						e1	ee	a1	ee	a2	ee	

(store) system state values at regular session. The deployment of freeze functions bytes (codes) in both MTU send request (message) and RTU send response (message) with security bytes (cryptography solution) are shows in Table 5

- The function bytes (codes) 0x000D, 0x000E, 0x000F, 0x0010, 0x00011 and 0x00012 are the application_control_functions bytes (codes). These functions are used to control the applications such as restart, initialization and start/shop. The deployment of application control functions bytes(codes) in both MTU send request (message) and RTU send response (message) with security bytes (cryptography solution) are shows in Table 6
- The function bytes (codes) 0x0013, 0x0014, 0x0015 and 0x0016 are the configuration_functions bytes (codes). These functions are used to configure the states (bytes state) such as save, enable/disable and class assignment while based on system behavior. Table 7 show the deployment of configuration functions bytes(codes) in both MTU send request (message) and RTU send response (message) with security bytes (cryptography solution)
- The function bytes (codes) 0x0017 and 0x0018 are the time_synchronization_functions bytes (codes). These functions are used to manage (measure) session within communication (between stations). Table 8 show the deployment of time synchronization functions bytes (codes) in both MTU send request (message) and RTU send response (message) with security bytes (cryptography solution)
- The function bytes (codes) 0x0019, 0x0001A, 0x001B, 0x001C, 0x001D and 0x001E are the file_functions bytes (codes). These functions are used to handle file states such as open, close, delete, get status, check security and abort. The function bytes (codes) from 0x0020 to 0x0080 are reserved and code 0x001F is uses for activate configuration (under development). Table 9 show the deployment of file synchronization functions bytes (codes) in

Table 9: File_functions and response_functions bytes with cryptography implementation

MTU/RTU Communication	Application Control (AC)	Function Code (FC)	Internal Indication (IIN)		Object field		Qualifier	Range		Encryption bytes						Padding bytes (optional)
0H0000(hex)			IIN_{1x}	IIN_{2x}	Group	Variation		Start	Stop							
Open file function																
MTU request	c3	19			46	03	5b	01		1e	ee	1a	ee	2a	ee	
RTU response	c3	81	00	00	46	04	5b	01		e1	ee	a1	ee	a2	ee	
Close file function																
MTU request	c3	1A			46	03	5b	01		1e	ee	1a	ee	2a	ee	
RTU response	c3	81	00	00	46	04	5b	01		e1	ee	a1	ee	a2	ee	
Delete file function																
MTU request	c3	1B			46	03	5b	01		1e	ee	1a	ee	2a	ee	
RTU response	c3	81	00	00	46	04	5b	01		e1	ee	a1	ee	a2	ee	
Get file information function																
MTU request	c3	1C			46	03	5b	01		1e	ee	1a	ee	2a	ee	
RTU response	c3	81	00	00	46	04	5b	01		e1	ee	a1	ee	a2	ee	
Authentication function																
MTU request	c3	1D			46	02	5b	01		1e	ee	1a	ee	2a	ee	
RTU response	c3	1D	00	00	46	04	5b	01		e1	ee	a1	ee	a2	ee	
Abort file function																
MTU request	c3	1E			46	03	5b	01		1e	ee	1a	ee	2a	ee	
RTU response	c3	81	00	00	46	04	5b	01		e1	ee	a1	ee	a2	ee	
Active configuration function																
MTU request	c3	1F								1e	ee	1a	ee	2a	ee	
RTU response	c3	81	00	00						e1	ee	a1	ee	a2	ee	
Function code:0×20(32)_0×80 (128): Reserved																
Function code: 0×81(129): Unsolicited response										1e	ee	1a	ee	2a	ee	

both MTU send request (message) and RTU send response (message) with security bytes (cryptography solution)

- The function bytes (codes) 0x0000 (confirmation), 0x0081 (response) and 0x0082(unsolicited response) are the response bytes (codes) and used within response message (remote station send response to master station). The response functions (codes) are also shows in Table 9

CONCLUSION

This study has been implemented a novel solution designated as "Dynamic Cryptography Buffer (DCB)" within Distributed Network Protocol (DNP3) stack while existing security solutions used external memory (space), during end-to-end implementations. In proposed study, the cryptography mechanism as additional security layer (or layers) is successfully deployed within DNP3 protocol stack and the performance results show that the "Dynamic Cryptography Buffer (DCB)" allocated space is sufficient for security implementation. This study gives new security trends to implement and test the cryptography algorithms as a potential security solution, within DNP3 protocol as a part of SCADA system.

In future work, the "dynamic cryptography buffer" will implement and test in other SCADA protocols (stacks) such as Modbus, Fieldbus and other, because the initial design of these protocols are also without any security concerned.

REFERENCES

DNP3, 2005. DNP3 specification volume 2: Application layer. Version 2.00 Draft K, October 8, 2005. http://www.docstoc.com/docs/139571815/dnp3-Application-Layer%E2%80%8E.pdf

Musa, S., A.A. Shahzad and A. Aborujilah, 2013a. Secure security model implementation for security services and related attacks base on end-to-end, application layer and data link layer security. Proceedings of the 7th International Conference on Ubiquitous Information Management and Communication, January 17-19, 2013, Kota Kinabalu, Malaysia.

Musa, S., A.A. Shahzad and A. Aborujilah, 2013b. Simulation base implementation for placement of security services in real time environment. Proceedings of the 7th International Conference on Ubiquitous Information Management and Communication, January 17-19, 2013, Kota Kinabalu, Malaysia.

Shahzad, A.A. and S. Musa, 2012. Cryptography and authentication placement to provide secure channel for SCADA communication. Int. J. Secur., 6: 28-44.

Shahzad, A.A., S. Musa, A. Aborujilah, M.N. Ismail and M. Irfan, 2013. Conceptual model of real time infrastructure within cloud computing environment. Int. J. Comput. Networks, 5: 18-24.

Shahzad, A.A., S. Musa, A. Aborujilah and M. Irfan, 2014a. A new cloud based supervisory control and data acquisition implementation to enhance the level of security using testbed. J. Comput. Sci., 10: 652-659.

Shahzad, A.A., S. Musa, A. Aborujilah and M. Irfan, 2014b. Industrial Control Systems (ICSs) Vulnerabilities analysis and SCADA security enhancement using testbed encryption. Proceedings of the 8th International Conference on Ubiquitous Information Management and Communication, January 8-10, 2014, ACM, New York, USA.