Prototyping a Lightweight Encryption on a Field Programmable Gate Array for Securing Tele-Control Data

Razali Jidin[1], Siti Noradhlia Tukijan[1], Ibrahim Al-Bahadly[2], Norziana Jamil[3], Qais Saif Qassim[3]
[1]*College of Engineering, Universiti Tenaga Nasional, Malaysia*
[2]*School of Engineering and Advanced Technology, College of Sciences, Massey University, New Zealand*
[3]*College of Computer Science & Information Technology, Universiti Tenaga Nasional*
razali@uniten.edu.my

***Abstract*- Financial and lives lost are possible consequences when disruption of an electricity supply occurs, due to security breaching of an industrial control system (ICS). Enhancing security of a geographical-wide ICS such as supervisory control and data acquisition (SCADA) system will require considerable efforts and capital. In addition to cost, system time sensitivity is also a major issue when one wants to implement security schemes at the existing SCADA. In addition to introduction on the need to secure SCADA, this paper is to focus on encryption for SCADA that are already in operation. Selecting a lightweight cipher such as lightweight encryption devices (LED) with its mathematical algorithms that suits hardware implementation, allows a complete cipher system to be hosted on a single low-cost FPGA, while satisfying time-latency of encrypting/decrypting of SCADA data packets. Though LED has no key scheduling and small key length, its security level is comparable to that of 256-bit Advanced Encryption Standard (256-AES), the current security adopted standard. Hardware architectures of LED for encrypting transmitted data are explored with insights on implementation of Galois multiplication into FPGA. In addition to provide a bump-in wire encryption, the proposed approach can be applied for Internet of Things (IoT).**

***Index Terms*— Lightweight cryptography; Securing SCADA; Lightweight Encryption Device; Field Programmable Logic Arrays; Galois field**

I. INTRODUCTION

Industrial Control Systems (ICSs) are considered as one of the most important types of Critical National Information Infrastructures (CNII) [1]. This is due to the fact that these systems are the backbones of many national infrastructures such as electricity generation, distribution and transmission. One of the most common examples of ICS is the Supervisory Control and Data Acquisition (SCADA) system. SCADA is considered as an event driven cyber physical system, encompasses centralized network with remotely located substations [2]. As of ICS, a typical SCADA system incorporates three main components: a control center, remote substations and communication networks that link all substations to the control center [3]. SCADA's control center is responsible for managing and supervising the overall system, processes and stores acquired information in addition to, in some cases, serves as a gateway to the corporate networks, which supports business operations. Due to the critical roles provided by the control center, any downtime or compromise of its processes can lead to disastrous consequences to the economy, public wellbeing and national security [4].

Prior to the Internet connectivity revolution in ICS/SCADA domain and cloud-based ICS/SCADA applications, SCADA systems were relatively isolated from other networks and external access [6]. Additionally, proprietary communication protocols and industrial devices were used which promote the security-by-obscurity concept. Therefore, cyber-security was never a key issue consideration; rather the issues were about developing a reliable, real-time and safe industrial control system. Nonetheless, today's technological trends and the advent of real-time information sharing and analysis have paved way for internetworking capabilities of the enterprise and business networks with the SCADA systems [6], [7]. This interconnectivity allows for more efficient remote control, management, and monitoring of industrial processes within the supervised system. The downside to this integration is that a large number of security threats have been introduced to the industrial domain.

In near future, electric vehicles can participate in the electrical grid frequency regulation. Such a scenario anticipates grids to be more intelligent, and sharing of communication networks with the public. Future energy system will be distributed compared to the present, such as a home energy system with photo-voltaic and inverters can be remotely controlled via Internet. Also Internet of Things (IoT) will be pervasive.

The future trend of grids with interconnected innumerable of IoT and other networks demands security measures to avoid cyber-attacks. Security measure to overcome cyber-attacks include authentication, encryption and integrity checks. Encryption is a process of converting a message or plaintext into an encrypted text. The encrypted message is supposedly able to be decoded only by the intended parties with a decryption key. The plaintext is encrypted using an encryption or a cipher algorithm with an encryption key. The process produces cipher-text that can only viewed in its original form if decrypted with the correct key. Examples of encryption algorithms are Advanced Encryption Standard 256-bit (AES-256), PRESENT, Lightweight Encryption Devices (LED) and PRINCE.

Implementing cipher system that support encryption and others on hardware on a single semiconductor chip such as

978-1-5386-6324-0/18/$31.00 ©2018 IEEE

Field Programmable Logic Arrays (FPGA) is to reduce both the computation time and energy consumption. The lightweight version cryptography algorithms have mathematical transforms that suitable for implementation on hardware. Different architectures of implementing LED on FPGA can be adopted to achieve contradicting constraints such as area size versus throughput. As an example to achieve smallest footprint of LED, authors in [21] had implemented the serial architecture that message is processed in 4-bit a time sequentially.

The rest of the paper is organized as follows: Section II reviews earlier research on securing SCADA, while Section III introduces lightweight block cipher cryptography, and LED algorithm is described in Section IV. Section V describes hardware architecture of LED with experimental results is in Section VI. Section VII concludes this paper.

II. SCADA TELE-CONTROL

A. Tele-control Data Links Protocol

An open standard protocol for SCADA tele-control information exchange over serial communication is defined by IEC 60870-5-101 protocol. The protocol architecture is defined by three OSI Reference Model layers: the Application Layer, the Data Link Layer and the Physical Layer. The Application Service Data Unit (ASDU) message is encapsulated inside the Data Link Layer frame or Link Protocol Data Unit (LPDU). The proposed implementation in this project is to encryption/decryption all data packets at the data link layer. One cipher is to be installed at each communication node, where encryption of packets receive from near node and decryption of packets coming from remote node.

B. Tele-control Encryption and Authentication

International standards bodies, such as Distributed Network Protocol (DNP) [11] and IEC which govern development of SCADA communication protocols have specified secure authentication extensions to their protocols. It has been proven that the strength of security correlates to the length of cryptographic keys. Longer keys yields better security due to higher processing complexity. However, the standards also caution users against using cryptography in time-sensitive SCADA networks due to low computational and low data rate capabilities of the devices.

Several researchers have proposed secure encryption for SCADA data transmission. Liu et al [12] proposed SEDEA a dynamic key generation method based on the underlying power system state estimator measurements. The encryption key varies synchronously with the power system dynamics and the key can be calculated in each remote terminal unit (RTU) of SCADA nodes.

Hadley et al [13] reported on the testing of the Secure SCADA Communication Protocol (SSCP).Their work was commercialized and incorporated as SEL-3241 products for serial cryptographic protection device. The SEL product is certified FIPS 140-3 Level 2. Tsang et al [14] published on low-latency a bump-in-the-wire cryptographic mechanism for SCADA protocols called YASIR that incorporates message prediction. Their solution guarantees data authenticity and freshness, and achieves low latency. However, the work did not specifically discuss YASIR's application to IEC 60870-5. Smith described cyber-threats to SCADA systems [15].

Earlier research on SCADA data security have not incorporated lightweight cryptography to meet time sensitive issue. In order to meet timeliness constraints of existing substation SCADA equipment, this work proposes lightweight cipher system hosted on a single hardware chip. Lightweight cryptography algorithms with less key-length and transforms that suit hardware implementation should able to meet execution time and small area footprint requirements.

III. LIGHTWEIGHT BLOCK CIPHERS

Lightweight block cipher is recent cryptography algorithms invented for devices that have limited computing power and power sources. The design criteria for lightweight cryptography are trade-off between security, hardware footprint, cost and performance.

The need for lightweight version algorithms has triggered invention of various ciphers such as PRESENT [20], PHOTON [21], LED [22], HIGHT [23], KLEIN [24], KATAN [25], SEA [26], TWINE [27] and LBlock [28]. In general, there are several critical aspects need to be considered in designing a block cipher, which include the round numbers, key scheduling scheme and encryption function [28]. LED is chosen for prototyping as without key scheduling, it is considered as ultra-lightweight, yet has security level comparable to 256 bit Advanced Encryption Standard (AES-256), the industry adopted encryption algorithm.

IV. LIGHTWEIGHT ENCRYPTION DEVICES

Light Encryption Device (LED) is one of the recent substitution-permutation network (SPN) lightweight block cipher, invented for small hardware footprint. The LED block cipher was conceived by Guo et al. [22] with 64-bit blocks of plain-text with variable key length of 64, 80, to 128-bit, to produce 64-bit cipher-text blocks.

Though LED is lightweight with short key length, and without key scheduling, it has sufficient security level, had been analyzed comparable to that of the 256-bit Advanced Encryption Standard (AES) previously [22, 28, 29, and 30].

A. LED Encryption

The block diagram for LED encryption is shown in Figure 1, that plain-text (PT) is processed in blocks of 64-bit with 64-bit key inputs to produce 64-bit cipher-text (CT) output blocks.

The first operation within LED is 64-bit addRoundKey (ak) operation between PT and sub key (SK_i). The i^{th} is for the number of round computation.

Following the first addRoundKey operation is round computation known as step, and then combination of addRoundKey and step is repeated (s-1) times. The value of s is either 8 or 12 for LED-64 and LED-128 respectively. The final operation is addRoundKey to produce a 64-bit cipher-text (CT) block.

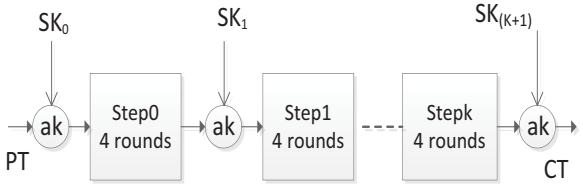


Figure 1: LED (Encryption) Block Diagram

Each step has four rounds that each round consists of four transforms named as Add-Constants (AC), Substitute-Cells (SC), Shift-Rows (SR) and Mix-Columns-Serial (MCS). For each of four rounds, different round constant (rc_x) are required by the AC as illustrated in Figure 2.

Figure 2: Each round requires different constant

For LED-64 encryption, the total step is eight (s = 8) with nine addRoundKey operations. As within each step there are four identical round transformations, therefore total round computation is 32. As the number of step for LED-128 is twelve, its number of round is 48. Therefore LED-128 requires 48 round constants.

B. AddRoundKey (ak)

As mentioned previously, Add-Key is 64-bit exclusive-or operation between the 64-bit PT block with 64-bit sub-key SK_i, that i^{th} indicates at which round. For LED-64, the SK_i, sub-keys as the as the following [17]:

$$Key = \begin{bmatrix} k_0 & k_1 & k_2 & k_3 \\ k_4 & k_5 & k_6 & k_7 \\ k_8 & k_9 & k_{10} & k_{11} \\ k_{12} & k_{13} & k_{14} & k_{15} \end{bmatrix}$$

Where $k_1 , k_2 , \ldots . k_{15}$ are to represent a value of a chosen key. For LED-64, the sub-key for every round is the same, the same secret value.

For LED-128, the sub-keys SKi for different rounds, KeyOdd and KeyEven are applied alternately to the addRoundKey. The alternate keys are as the following [17]:

$$KeyOdd = \begin{bmatrix} k_0 & k_1 & k_2 & k_3 \\ k_4 & k_5 & k_6 & k_7 \\ k_8 & k_9 & k_{10} & k_{11} \\ k_{12} & k_{13} & k_{14} & k_{15} \end{bmatrix}$$

$$KeyEven = \begin{bmatrix} k_{16} & k_{17} & k_{18} & k_{19} \\ k_{20} & k_{21} & k_{22} & k_{23} \\ k_{24} & k_{25} & k_{26} & k_{27} \\ k_{28} & k_{29} & k_{30} & k_{31} \end{bmatrix}$$

The KeyEven and KeyOdd are derived as the following:

SKi(j) = k(j + i *16 mod L);

Where

L is number of nibble = 16,

j is position in the matrix.

All k(j) within each matrix to represent the secret or chosen key

C. Round Computation

The plain-text message PT of 64-bit or arrange as sixteen 4-bit nibbles block: $m_0 \| m_1 \| m_2 \ldots . m_{14} \| m_{15}$ is converted to a matrix with sixteen elements that each element is 4-bit nibble as the following format:

$$64\ bit\ PT = \begin{bmatrix} m_0 & m_1 & m_2 & m_3 \\ m_4 & m_5 & m_6 & m_7 \\ m_8 & m_9 & m_{10} & m_{11} \\ m_{12} & m_{13} & m_{14} & m_{15} \end{bmatrix}$$

This plain-text matrix is considered as an initial cipher state. The initial state is then fed to LED for transformation of steps after the addRoundKey operation. For LED64, there will 8 steps and 9 addRoundKey operations with sequential programming code in Table 1.

As mentioned previously for LED-64, the sub-keys for the addRoundKey is constant, equals to $(64)_{10} = (40)_{16}$, while add-constant changes every round.

Table 1: LED64 Encryption Pseudo Code

```
s = 8 --- for LED64
state= convert(PlainText64-to-4x4Matrix)  -- initial state

for i = 0 to (s-1) do {

    state =   addRoundKey(state, SKi)

    for j = 0 to 3 do {              -- one step = 4 rounds
        state = AC (state, rc (i, j))
        state =  SC (state)
        state =  SR (state)
        state =  MCS(state)
        };
    };                                   -- total 8 steps

state = addRoundKey( state, SK8)

CipherText64 = 4x4matrix-to-array(state)
```

D. Add Constant (AC)

The operation within AC is exclusive-OR operation of first two columns of matrix with round-dependent constants called as round constant (rc) as given below [17]:

For every round, the round constant of 6-bit lengths is shifted to the left by one position with the least significant bit, rc_0 is being feedback with a function of $rc_5 \oplus rc_4 \oplus 1$. The values of rc_i are represented in bytes.

$$\begin{bmatrix} 0 \oplus (ks_7 \parallel ks_6 \parallel ks_5 \parallel ks_4) & (rc_5 \parallel rc_4 \parallel rc_3) & 0 & 0 \\ 1 \oplus (ks_7 \parallel ks_6 \parallel ks_5 \parallel ks_4) & (rc_2 \parallel rc_1 \parallel rc_0) & 0 & 0 \\ 2 \oplus (ks_3 \parallel ks_2 \parallel ks_1 \parallel ks_0) & (rc_5 \parallel rc_4 \parallel rc_3) & 0 & 0 \\ 3 \oplus (ks_3 \parallel ks_2 \parallel ks_1 \parallel ks_0) & (rc_2 \parallel rc_1 \parallel rc_0) & 0 & 0 \end{bmatrix}$$

For LED-64 that has a total 32 rounds of computation require 32 round constants. The value of $(ks_7, ks_6, \ldots ks_0)$ is an 8 bits key size representation, with ks_0 being the least significant bit (LSB). The round constants generated by a Linear Feedback Shift Register (LFSR).

E. Substitution Cell (AC)

Each nibble element of the matrix is replaced by the nibble produced by a table called S-Box. Parts of S-Box for LED is given in Table II (x is replaced by S(x)), is similar to that of PRESENT cipher [14]. The SubCells is to minimize the correlation between input and output.

Table 2: Substitute-Cell [20]

x	0	1	2	3	4	5	6	7
S(x)	C	5	6	B	9	0	A	D

F. Shift Rows (SR)

Shift Rows operates on the rows of the matrix or state, is similar to shift operation that of earlier developed encryption known as AES. The nibble of every row is rotated to the left by i^{th} position, for i=0,1,2, and 3.

G. Mix-Columns Serial (MCS)

Mix-Column-Serial function takes each column vector of the array state and update with the new column vector after multiplying the vector by a fixed constant matrix M, the Maximum Distance Separable (MDS) [17]. The arithmetic employed for this MCS operation is finite field arithmetic known as Galois Field (GF) with an irreducible polynomial of x^4+x+1.

$$M = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 4 & 1 & 2 & 2 \end{pmatrix}^4 = \begin{pmatrix} 4 & 1 & 2 & 2 \\ 8 & 6 & 5 & 6 \\ B & E & A & 9 \\ 2 & 2 & F & B \end{pmatrix}$$

For MCS, the finite field is GF (2^4) with irreducible polynomial of x^4+x+1. Definition for GF (p^n) is its elements represented as polynomials with degree less than n, where p is a prime number, and n is positive integer. For mathematical operation such as multiplication, typical multiplication of integer then followed reduction by modulo p or compute reminder of modulo R, where R is an irreducible polynomial. The implementation of multiplication in finite field of Galois on FPGA will be presented in the next section namely the hardware architecture.

V. LED HARDWARE ARCHITECTURES

Choices of hardware architectures to realized LED into hardware include pipelining and serial. The following subsections describe options of architecture available and details hardware implementation of LED's round computations.

A. LED-64 Hardware Architectures

Serial architecture of LED as mentioned in [21], being smallest footprint that process message sequentially requires high number of clock cycles. The totals clock cycle are 1248 (39 cycles per round computation) and 1872 for LED-64 and LED-128 respectively.

Alternatively, one can implement one round computation as a component that transforms the LED states repeatedly. Therefore for LED-64, the 32 round computations require 32 clock cycles, starting with plain-text as initial state input. For this architecture, additional components are a 64-bit register, state machine and a component that selects the round constants. Other approach is using pipeline architecture such as in [31].

To reduce clock cycle, one can implement the whole 32 round computations of LED-64 encryption as a combinational logic within an FPGA. This type of implementation consisted of 32 components of each addRoundKey, AddConstant, Substitute Cell, ShiftRow and MixColumnSerial.

In this project, combinational logic architecture has been chosen to achieve the least execution time. All components have been coded in hardware descriptive language (HDL). The snippet of VHDL code for the step with AddRoundKey (AddRndKey), and each step "instantiation" of OneStep is as shown in Table 3.

Table 3: Snippet of HDL Code for LED-64

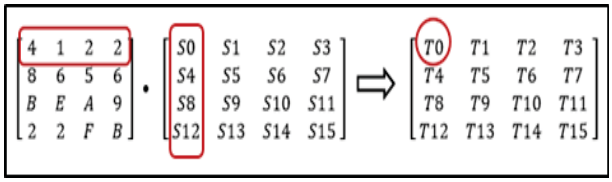
```
AddRndKey1: PT0 <= Key64 xor PlainText;
step1 : Entity work.OneStep port map (0, PT0, CT0);

AddRndKey2: PT1 <= Key64 xor CT0;
step2 : Entity work.OneStep port map (1, PT1, CT1);
……
AddRndKey7: PT6 <= Key64 xor CT6;
step7 : Entity work.OneStep port map (1, PT6, CT6);
```

OneStep performs four rounds of transformation that each round has AC, SC, SR and MCS component. As each step has four round computation, there are four components of AC, SC, SR and MCS respectively within the OneStep. Therefore total component let say for AC is 32.The final state is converted back to 64-bit format, to become the cipher-text.

B. Mix Column Serial Architecture

The mix-column serial operation of LED is basically multiplication of two matrices (each element is 4-bit nibble) as the following:



$$T0 = (4 \bullet S0) \oplus (1 \bullet S4) \oplus (2 \bullet S8) \oplus (2 \bullet S12)$$
$$T1 = (4 \bullet S1) \oplus (1 \bullet S5) \oplus (2 \bullet S9) \oplus (2 \bullet S13)$$
$$T2 = (4 \bullet S2) \oplus (1 \bullet S6) \oplus (2 \bullet S10) \oplus (2 \bullet S14)$$

Matrix MDS multiply to nibbles S0, S1, S2, and so on until S15. The result of multiplication is represented T0, T1 and so on. For each element let say T0 should require four multiplications of two 4-bit operands.

Table 4: Snippet of HDL Code for Matrix GF (2^4) Multiplication

```
--GF: Multiplication & Reduction to Produce 4-bit result

M   : std_logic_vector(3 downto 0); -- MDS matrix element
S   : std_logic_vector(3 downto 0); -- State matrix element
……..

--POLY = X^4 + X + 1
poly       :  std_logic_vector(4 downto 0) := "10011";
…….

-- MULTIPLICATION    res <= M x S;
res(0) <=  M(0) and PT(0);
res(1) <= (M(1) and PT(0)) xor  ((M(0) AND PT(1));
……..
res(6) <=  M(3) and PT(3);
……

-- CHECKING BIT 6 if magnitude greater than POLY
 ……..

-- CHECKING BIT 5 if magnitude greater than POLY
…….

-- CHECKING BIT 4 if magnitude greater than POLY

T <= ….   ; -- irreducible result
```

For each multiplication of two 4-bit operands, conventional multiplication of 4-bit operation with division by polynomial X^4+X+1 to produce irreducible result within the Galois field values are required. However, rather than performing multiplications, AND with EXCLUSIVE-OR operations have been used to produce product res(0), res(1), res(2), ….. res(6). Similarly, instead of performing division to reduce multiplication product to 4-bit, res(6) is checked if equal to logic '1', if so, EXCLUSIVE-OR with "1001100" is performed to produce new result. Subsequently bit 5 of new result is checked equal to logic '1', if true, perform EXCLUSIVE-OR with "010010", to produce latest result, and so on for bit 4 of latest result. Subsequently, the finite product is derived from 4-bit of bit 4 manipulation result. The HDL code for described logical operations to realize multiplication and division to reduce result to 4 bit Galois field finite value is listed in Table 4. Summation of products using exclusive-or operations yield the T(i), where i is 0,1,2, …..15.

VI. EXPERIMENTAL RESULTS

A. *Correctness Test*

LED-64 encryption has been implemented on Spartan-6 SP601 XC6SLX16-2CSG324 FPGA. Two test vectors (64-bit plaintext and 64-bit key), to evaluate LED encryption correctness, has produced the following cipher texts.

Plaintext1 (PT): x"0000_0000_0000_0000"
Key: x"0000_0000_0000_0000"
Ciphertext1 (CT): x"897C_0A30_0104_2C93"

Plaintext2 (PT): x"FEDC_BA98_7654_3210"
Key: x"FEDC_BA98_7654_3210"
Ciphertext2 (CT): x"85CF_3983_E155_300A"

The correctness has been validated by two methods: simulation test and also downloading onto an FPGA on an evaluation board for observe encrypted text on a display.

B. *Synthesis Report*

The synthesis report of LED encryption module is as shown in Table 5. The number of look-up-table (LUT) is 96 out of 9,112, which is only one percent of total LUT on SPARTAN 6.

Table 5: LED-64 Encryption Synthesis Report

Device	**Spartan-6 XC6SLX16-2CSG324**
Number of Slice Registers	104 out of 18,224
Number of Slice LUTs	96 out of 9,112
Number used as logic	95 out of 9,112
Number of LUT Flip-Flop	99
Maximum Frequency	266.238MHz

Though the biggest footprint architecture has been selected for implementation, the number of FPGA resources used such as LUT is 96 out 9,112 slices (1.05% of SPARTAN-6). Such low footprint allows the FPGA to host others such as processors for IoT.

C. *LED-64 Cipher System for SCADA IEC101 Data*

Figure 5 illustrates the LED-64 with two asynchronous receive-and-transmits, two buffers and a controller that have been realized on SPARTAN 3E FPGA. The set-up is to evaluate LED-64 as a crypto-system to encryption/decrypt serial data streams, as an initial prototype. The buffers are for data buffering that to adapt data throughput and data-width between serial and crypto-algorithm processing. Each buffer has two clock sources; one is the system clock (sys_clk) and the other is clock generated by the UART (uart_clk). The uart_clk has frequency multiple of serial data baud rates to synchronize data movement in/out of buffers.

The controller is to detect data availability in the buffers, to initiate an encryption process, and also to check

completion of encryption.

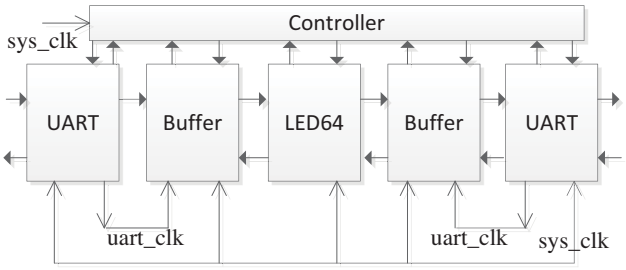


Figure 3: LED-64 Cipher System

The baud rate for serial communication can be adjusted to 9600 to 115K. Samples of SCADA payload (of IEC 60870-5-101) for encryption are tabulated in Table 6.

Table 6: Samples Payload Encrypted/Decrypted

APDU type	Function (Control/Monitor)	APDU Length	Payload
U-format	Start data transfer	4	68 04 07 00 00 00
U-format	Test (control)	4	68 04 43 00 00 00
U-format	Test (monitor)	4	68 04 83 00 00 00
I-format	Command	14	68 0e 1a 00 ……
I-format	Value short float	250	68 fa 00 …….

VII. Conclusion

Related research on the need to securing SCADA data and revision of lightweight encryption devices (LED) algorithm provide background for this project. Several architectures of LED implementation into hardware have been reviewed, and followed by a detail design of components within LED on FPGA. In this project, hosting a complete LED-based cipher system on a single FPGA offers a low-cost security device for legacy SCADA, while able to meet timelines requirement. Evaluation of a selected cipher, LED in this case, has been successfully implemented on a low-cost FPGA to encrypt tele-control data packets. The size of synthesized LED-64 algorithm requires merely one percent of low cost FPGA resources (SPARTAN 6), and yet able to process data packets at 266MHz.

Acknowledgment

The authors would like to extend their gratitude to Ministry of Higher Education Malaysia (MOHE) for their FRGS grant no: FRGS/1/2015/ICT06/UNITEN/02/1.

References

[1] C. Alcaraz and S. Zeadally, “Critical infrastructure protection: Requirements and challenges for the 21st century,” Int. J. Crit. Infrastruct. Prot., vol. 8, pp. 53–66, Jan. 2015.

[2] L. A. Maglaras et al., “Cyber security of critical infrastructures,” ICT Express, vol. 4, no. 1, pp. 42–45, Mar. 2018.

[3] B. Miller and D. Rowe, “A survey SCADA of and critical infrastructure incidents,” in Proceedings of the 1st Annual conference on Research in information technology - RIIT ’12, 2012, p. 51.

[4] S. Nazir, S. Patel, and D. Patel, “Assessing and augmenting SCADA cyber security: A survey of techniques,” Computer. Security. vol. 70, pp. 436–454, Sep. 2017.

[5] R. S. Ramachandruni and P. Poornachandran, “Detecting the network attack vectors on SCADA systems,” 2015 Int. Conf. Adv. Computer Communication Informatics, ICACCI 2015, pp. 707–712, 2015.

[6] U. P. D. Ani, H. (Mary) He, and A. Tiwari, “Review of cybersecurity issues in Industrial Critical Infrastructures: Manufacturing in perspective,” J. Cyber Secur. Technol., vol. 1, no. 1, pp. 32–74, Jan. 2017.

[7] M. Krotofil and D. Gollmann, “Industrial control systems security: What is happening?,” in 2013 11th IEEE International Conference on Industrial Informatics (INDIN), 2013, pp. 664–669.

[8] R. Tawde, A. Nivangune, and M. Sankhe, “Cyber security in smart grid SCADA automation systems,” in 2015 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), 2015, pp. 1–5.

[9] W. Knowles, D. Prince, D. Hutchison, J. F. P. Disso, and K. Jones, “A survey of cyber security management in industrial control systems,” Int. J. Crit. Infrastruct. Prot., vol. 9, pp. 52–80, Jun. 2015.

[10] Wei Gao, T. Morris, B. Reaves, and D. Richey, “On SCADA control system command and response injection and intrusion detection,” in 2010 eCrime Researchers Summit, 2010, pp. 1–9.

[11] Gilchrist, G. (2008, July). Secure authentication for DNP3. In Power and Energy Society General Meeting-Conversion and Delivery of Electrical Energy in the 21st Century, 2008 IEEE (pp. 1-3). IEEE Chicago

[12] Liu, T., Tian, J., Gui, Y., Liu, Y., & Liu, P. (2017). “SEDEA: State Estimation-Based Dynamic Encryption and Authentication in Smart Grid”. IEEE Access, 5, 15682-15693.

[13] Hadley, M. D., Huston, K. A., & Edgar, T. W. (2007). AGA-12, Part 2 Performance Test Results. Pacific Northwest National Laboratories. Chicago

[14] Tsang, Patrick P., and Sean W. Smith. "YASIR: A low-latency, high-integrity security retrofit for legacy SCADA systems." IFIP International Information Security Conference. Springer, Boston, MA, 2008.

[15] Security for Critical Infrastructure SCADA Systems, SANS Institute, A. H. Smith, Feb 2005.

[16] IEC 60870-5-101:2003, “Telecontrol Equipment and Systems – Part 5-101: Transmission protocols – Companion standard for basic telecontrol tasks”

[17] IEC 62351-5:2009, “Power system management and associated information exchange – Data and communication security – Part 5: Security for IEC 60870-5 and derivatives

[18] M. Mozaffari-Kermani, K. Tian, R. Azarderakhsh, and S. Bayat-Sarmadi, “Fault-resilient lightweight cryptographic block ciphers for secure embedded systems,” IEEE Embed. Syst. Lett., vol. 6, no. 4, pp. 89–92, 2014.

[19] A. Bogdanov et al., “PRESENT: An Ultra-Lightweight Block Cipher,” Cryptogr. Hardw. Embed. Syst. - CHES 2007, pp. 450–466, 2007.

[20] J. Guo, T. Peyrin, and A. Poschmann, “The PHOTON Lightweight Hash Functions Family,” Crypto, pp. 222–239, 2000.

[21] J. Guo, T. Peyrin, A. Poschmann, and M. Robshaw, “The LED block cipher,” Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 6917 LNCS, pp. 326–341, 2011.

[22] D. Hong et al., “HIGHT: A New Block Cipher Suitable for Low-Resource Device,” Cryptography Hardware Embed. Syst. CHES 2006, Springer, LNCS, vol. 4249, pp. 46–59, 2006.

[23] Z. Gong, S. Nikova, and Y. W. Law, “KLEIN: A New Family of Lightweight Block Ciphers,” pp. 1–18, 2012.

[24] C. De Cannière, O. Dunkelman, and M. Knežević, “KATAN and KTANTAN - A family of small and efficient hardware-oriented block ciphers,” Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 5747 LNCS, pp. 272–288, 2009.

[25] G. Piret and N. Gershenfeld, “SEA for internet-0 : a Scalable Encryption Algorithm for Small Embedded Applications why we need crypto for internet-0,” pp. 222–236.

[26] T. Suzaki, K. Minematsu, S. Morioka, and E. Kobayashi, “TWINE : A Lightweight Block Cipher,” no. 1, pp. 339–354, 2013.

[27] W. Wu and L. Zhang, “LBlock: A Lightweight Block Cipher,” pp. 327–344, 2011.

[28] S. Salim, M. Aldabbagh, and I. Al, “Lightweight Block Ciphers : a Comparative Study,” vol. 2, no. 4, pp. 159–165, 2012.

[29] M. Fujishiro, M. Yanagisawa, and N. Togawa, “Scan-based Attack on the LED Block Cipher Using Scan Signatures,” pp. 1460–1463, 2014.

[30] W. Li et al., “Impossible Differential Fault Analysis on the LED Lightweight Cryptosystem in the Vehicular Ad-Hoc Networks,” IEEE Trans. Dependable Secur. Comput., vol. 13, no. 1, pp. 84–92, 2016.

[31] P. F. R. Sofia, T. B. Sheeba, and E. M. K. Engineering, “Efficient Implementation of (LED) Light Encryption Device Using Pipeline Architecture,” vol. 24, pp. 173–176, 2016