Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 143 (2018) 736–743

Procedia Computer Science

www.elsevier.com/locate/procedia

8th International Conference on Advances in Computing and Communication (ICACC-2018)

High Speed and Low Power Implementation of AES for Wireless Sensor Networks

Sreenath Thangarajan[a], V S Kanchana Bhaaskaran[a,]*

[a]*School of Electronics Engineering, Vellore Institute of Technology, Chennai – 600063, India.*

Abstract

In the recent years, data security has become the biggest concern due to the increasing number of connected devices. Hence, cryptography has become vital for enhancing data security. Cryptography is a technique which converts the data into an unintelligible form. In applications such as the wireless sensor networks, it plays a major role since most of the data is transmitted over an insecure channel. Symmetric key cryptosystems play a major role in such applications, since they are lightweight and faster in operation. Power dissipation of the system is another major concern for such applications as they are battery-operated devices. In this paper, the power dissipation of the circuit is enhanced by trading off area and throughput. The power is minimized by the method of parallel processing the hardware along with reduced amount of redundant hardware. The power dissipation of the circuit for the proposed structure is presented to be 2.04 times less than that of existing parallel processing structures. The proposed architecture was implemented using the industry standard Cadence® Encounter SoC tools using TSMC180 technology library.

Keywords: Cryptography; AES; Pipelining Architectures; Low Power Cryptography Design; High-Speed Cryptography Circuits.

* Corresponding author. Tel.: +91-735-878-2584.
E-mail address: vskanchana@ieee.org

1877-0509
10.1016/j.procs.2018.10.440

1. Introduction

With the developments in networking technology, the need for automated tools for protecting the data stored on a personal computer from vulnerabilities and attacks have become mandatory. Such vulnerabilities are more prominent in devices, which are networked through public switched telephone networks (PSTN) and time-sharing systems. Furthermore, the developments in distributed computing require a high degree of trust between the administrator and the terminal user, and thus it is extremely important to protect the data in transit.

Cryptography is a science that applies logic and complex mathematics to design robust techniques to encrypt and protect the data [1]. The two common classifications of the cryptographic techniques are the public key (asymmetric) and the private key (symmetric) cryptographies. The public key cryptosystem uses two different keys for encryption and decryption while the private key cryptosystems use the same key for encryption and decryption. The main advantage with the public key cryptosystem is that it can be used to transfer data with unknown entity without any prior information about the entity due to the use of two separate keys for encryption and decryption. On the other hand, in the private key cryptosystems, the key has to be shared over a secure channel for unknown entities or the key is previously either stored or shared in the case of known entities. The main advantage of private key cryptosystems over public key cryptosystems is that the public key cryptosystems require a lot of computation power and has a longer computation time. Most of the private key cryptosystem, viz. Data Encryption Standard (DES) [2] or Advanced Encryption Standard (AES) [3] is lighter than the private key cryptosystems like Rivest, Shamir and Adelman algorithm (RSA) or Elliptic Curve Cryptography (ECC). Advanced Encryption Standard (AES) is less vulnerable to all the known attacks. AES uses some of the most common techniques employed in cryptography such as Substitution and Transposition sequentially to encrypt the data. There are three versions of AES, namely, AES-128 (10 rounds and 128-bit Key), AES-192 (12 rounds and 192-bit Key) and AES-256 (14 rounds and 256-bit Key).

The Wireless Sensor Networks (WSN) has emerged as a major source of information in various applications such as the healthcare, remote sensing, weather monitoring, etc. In general, the WSNs are limited in terms of power, computational efficiency and communication [4]. Furthermore, they are deployed in inaccessible areas and this makes servicing of the devices and replacing the battery of devices becomes almost impossible. The data obtained from such sensors need to be very accurate and reliable to devise prediction models. However, their limited computational power for encrypting the data from the sensors is a matter of concern. Hence, the data from the WSNs must be encrypted using secure lightweight cryptographic algorithms without compromising on the power dissipation and throughput. Though the AES is secure and lighter than other public key cryptosystems, the power dissipation is relatively high, which makes it less practical to use in the nodes of networked devices [5].

The power dissipation of a system can be optimized by trading off against the throughput, since the WSN systems usually have lesser throughput. References [6] [7] minimize the power dissipation by optimizing the S-Box and Key memory of the design. It is achieved by replacing the LUT based S-Box with Rijndael's Galois Field based S-Box. In [8] [9], the throughput of the system is increased without considering the power dissipation of the circuit. The proposed architecture minimizes the power dissipation using techniques mentioned in [10] and by additionally considering 1) parallel processing and 2) trading off the throughput.

In this paper, the implementation techniques of AES and techniques to minimize the power consumption for high-speed processes is presented. Section 2 briefs the theoretical background of the AES algorithm. The implementation schemes of the traditional AES are depicted in Section 3, while Section 4discusses its low power implementation schemes. Section 5 presents the results of the traditional AES and low power optimized AES, and comparison is made between them. Section 6 concludes.

2. Advanced Encryption Standard

The AES is a cryptographic algorithm specified and approved by the Federal Information Publishing Standards (FIPS). It is a symmetric block cipher algorithm that can encrypt data to an unintelligible form (called cipher text) and decrypt information to retrieve the data from the cipher text. In the AES algorithm, all the bytes are interpreted as finite field elements. Addition and subtraction in finite fields are performed by modulo-2 addition, while the multiplication is performed with the help of an irreducible polynomial over the GF (2^8). The irreducible polynomial for the chosen Galois Field is

$$m(x) = x^8 + x^4 + x^3 + x + 1 \quad (1)$$

For AES algorithm in general, the length of the state, input block, and output block is 128 bits, which are represented by $N_b = 4$. This represents that there are four rows of 32-bit words in each block. The key length of the AES-128, 192 and 256 are 128-bits, 192-bits and 256-bits, respectively. There is no restriction on key selection as there are no weak or semi-weak keys identified. For both encryption and decryption, AES uses four different byte-oriented transformations as shown in Fig. 1, namely, 1) Byte substitution using S-Box, 2) shifting of rows by different offsets, 3) mixing data within each column and 4) adding round key to each state. Each transformation process is explained in detail as follows.

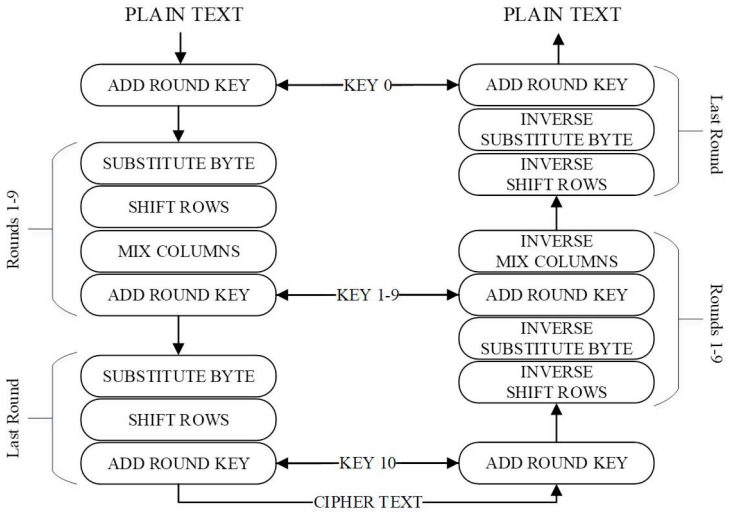

Fig. 1. Advanced Encryption Standard Algorithm [3]

2.1. Substituting Bytes

The substitute bytes transformation [3] is a non-linear byte substitution technique that operates on each byte independently. For instance, to map *8'h00* to itself, the multiplicative inverse of the element is taken and then the affine transform is applied to it over the GF(2) which results in *8'h63*. The S-Box technique thus obtained is invertible and can be directly used for inverse S-Box for decryption.

2.2. Shifting of Rows

The bytes in the last three rows of the block are shifted cyclically in each round by a different number of offsets [3]. The first row is not shifted while only the other rows are shifted. A similar process can be carried out for decryption process by shifting the rows cyclically through complementary offsets.

2.3. Mixing of Columns

Each column in the state is considered as a four-term polynomial on which each transformation is applied column-by-column [3]. Over the Galois Field (2^8), the columns are multiplied with $x^4 + 1$ with a fixed polynomial size a(x). The operation is performed in all the rounds except the last round. Similarly, in the decryption process, the mix-columns operation is performed in all the rounds except the first round.

2.4. Adding Round Key

A round key [3] is generated in each round of the algorithm with respect to the initial key. The key is added (modulo-2) to the data at the end of each round. Based on the current output of the S-Box, a new key is generated at each stage, which is then added to the data at that stage.

3. Implementation of the Conventional AES Algorithm

The major blocks for implementing the AES Algorithm were identified as Key memory, encipher block, decipher block and S-Box/Inverse S-Box. The AES algorithm has been implemented using Finite State machines and each block is explained individually as follows:

S-Box / Inverse S-Box

The S-Box was implemented in the form of a Look-up Table. The value of each block in the S-Box will not change for each iteration as it changes for Key memory. Similarly, inverse S-Box is also implemented using Lookup Table [6]. The Look-up Tables for the S-Box and inverse S-Box are shown in Table 1 and Table 2, respectively. The S-Box memory is designed in such a way that the data in the look-up table can be accessed in parallel by five different multiplexers, of which four multiplexers are accessed by the input data and the fifth multiplexer is accessed by the Key memory. This ensures that all the bytes in a round are substituted at the same time instant.

TABLE I. S-BOX LOOK-UP TABLE

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
30	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
40	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a0	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

TABLE II. INVERSE S-BOX LOOK-UP TABLE

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
10	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
20	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
30	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
40	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
50	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
60	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
70	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
80	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
90	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
a0	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
b0	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c0	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
d0	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
e0	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
f0	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Key Memory

The AES key memory is a round key generator [8], which generates a new key for each round. The logic for generating a round key is shown below.

$trw = sub\_bytes\ (rotate\_word\ (w_i[3]))\ \hat{}\ RCON$

$w_{i+1}[0] = w_i[0]\ \hat{}\ trw$

$w_{i+1}[1] = w_i[1]\ \hat{}\ w_i[0]\ \hat{}\ trw$

$w_{i+1}[2] = w_i[2]\ \hat{}\ w_i[1]\ \hat{}\ w_i[0]\ \hat{}\ trw$

$w_{i+1}[3] = w_i[3]\ \hat{}\ w_i[2]\ \hat{}\ w_i[1]\ \hat{}\ w_i[0]\ \hat{}\ trw$

Here, the RCON register stores the following values in it. RCON = {*8'h01, 8'h02, 8'h04, 8'h08, 8'h10, 8'h20, 8'h40, 8'h80, 8'h1B, 8'h36*}. The value from the RCON is selected depending on the rounds of the AES.

Encipher Block

The Encipher block consists of substitute bytes, shift rows, mix columns and add round keys blocks as combinational blocks. The round keys are obtained from the key memory at the end of each stage of the round. The mix columns step is not included in the final round of the FSM while all the other combinational blocks are included in each round.

Decipher Block

The Decipher block consists of inverse substitute bytes, inverse shift rows, inverse mix columns and add round keys blocks as combinational blocks. The round keys are obtained from the key memory at the end of each stage of the round. The inverse mix columns is not included in the last round of the FSM while all the other combinational blocks are included in each round.

4. Low Power Implementation

For high-speed applications, the power dissipation in the design can become very high since the frequency of operation is very high. The frequency of operation cannot be minimized as it affects the performance of the system. Thus, by trading off with the area and computational complexity, the power consumption of the design is minimized using the following techniques.

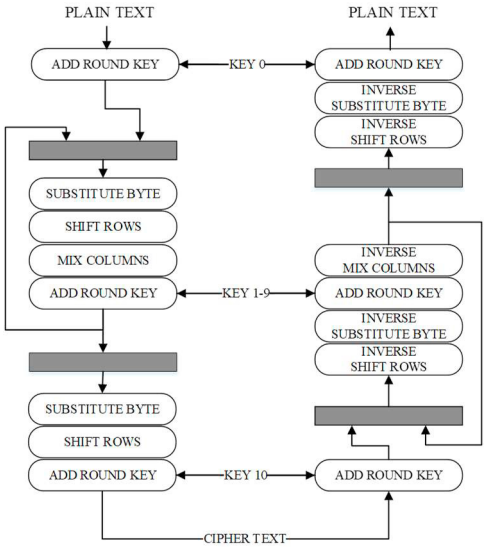

Fig. 2. AES – without parallel processing[3]

For instance, if the AES is implemented using a single hardware as shown in Fig. 2, then to get two cipher texts at the output, 20 machine cycles are required. On the other hand, if the system is parallel processed by replicating the hardware 10 times as shown in Fig. 3, then during the same 20 machine cycles, 10 cipher texts can be obtained.

4.1. Rijndael's S-Box

The Look-up Table based architectures are efficient only for FPGA based designs. However, for semi-custom based design, the use of Look-up Tables (LUTs) will increase the area overhead and increase power dissipation. The S-Box for encryption process is implemented using Rijndael's S-Box by

1. Calculating the multiplicative inverse of the input with respect to the irreducible polynomial over the GF(2^8)
2. Applying affine matrix transformation to the output of step 1 to obtain the substituted value for the input.

A similar process is applied for computing the inverse S-Box value for a particular input, which is used in the decryption process.

4.2. Parallel Processing in AES Algorithm

Parallel Processing in AES can be obtained by replicating the hardware and synchronizing it. The hardware is replicated 10 times for AES-128 since it has 10 rounds. The parallel processing can be used to either increase the throughput of the process or decrease the power dissipation. Though the throughput is increased or power dissipation is reduced by decreasing the frequency, the area overhead has drastically increased to 10 times. Thus, there has to be a trade-off made between area and power dissipation/throughput.

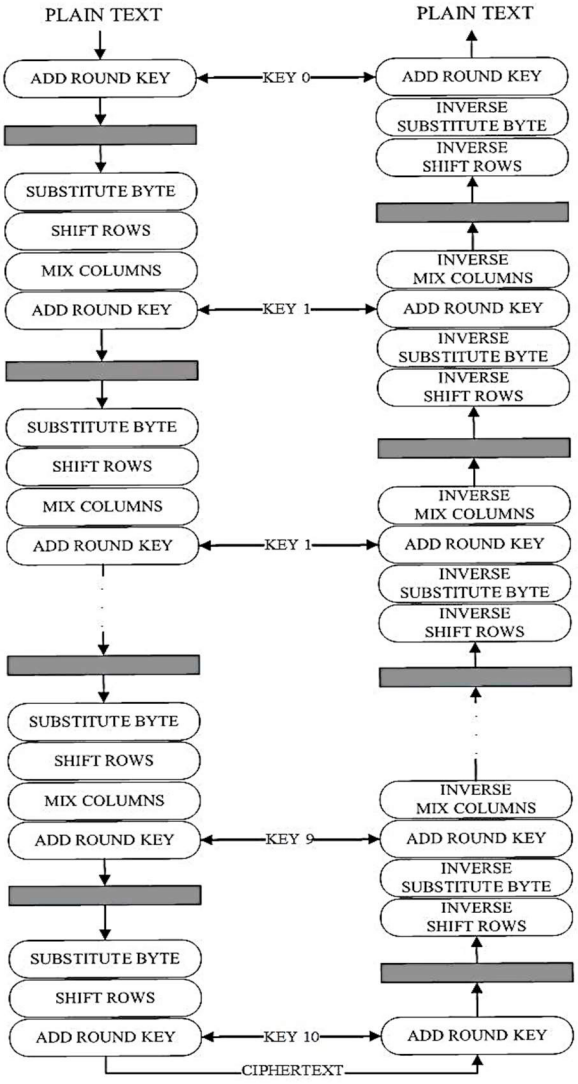

Fig. 3. AES – Parallel Processing with order 10 [9]

In order to optimize the design, the hardware is replicated three times and the final stage of AES have been implemented separately. Fig. 4 shows the detailed implementation of encryption and decryption of the AES algorithm using the optimized structure. For the implementation given in Fig. 4, in the same 20 machine cycles 5 cipher texts can be obtained for the given plain text. The area of the system is increased four times, since only four instances of the hardware have been implemented thus increasing the throughput or decreasing the power dissipation.

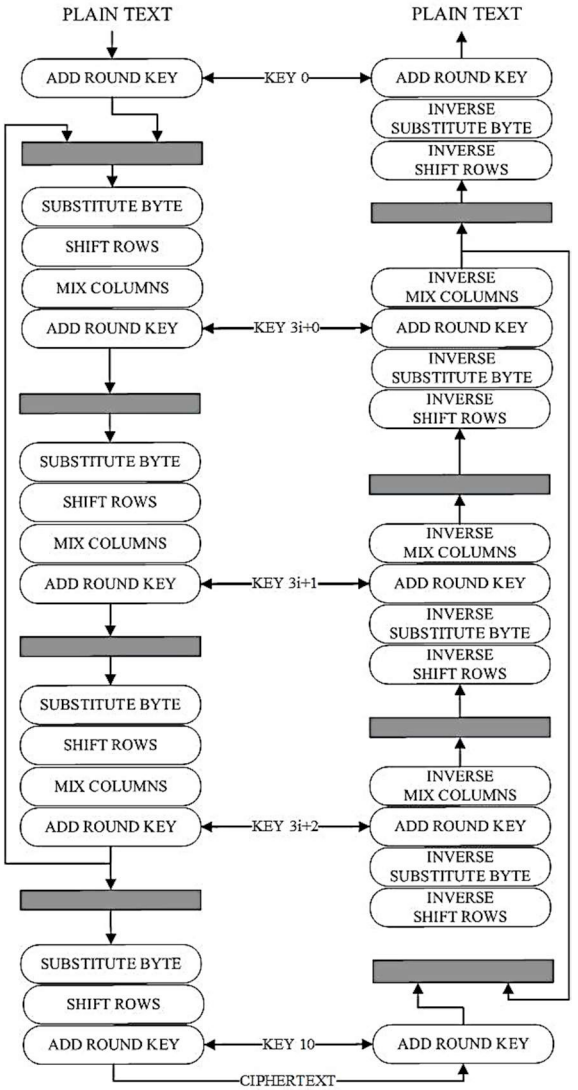

Fig. 4. Proposed Architecture – AES – Parallel processing with order 3

5. Results and Discussions

The architecture shown in Fig. 4 was implemented and, the area and power dissipation were compared with the architectures shown in Fig. 2 and Fig. 3. The power dissipation and area components of the hardware have been found to be increasing as the order of parallel processing increases. From Fig. 5 (a), it can be observed that in parallel processing with order 10, as the throughput increases, the power dissipation of the circuit rises three times than that of the power dissipation of first-order parallel processing. The power dissipation of the first order, third order, and tenth order parallel processing structures were found to be 39.786 mW, 57.499 mW and 118.986 mW respectively, while the throughput of the process is 1, 5 and 10 cipher texts for 20 machine cycles.

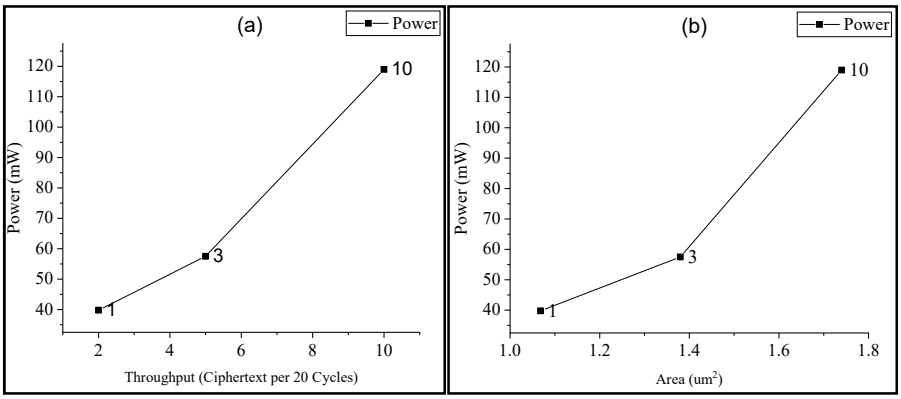

Fig. 5. (a) Power Dissipation and Throughput w.r.t order of Parallel Processing. (b) Power Dissipation and Area w.r.t order of Parallel Processing.

Another advantage of having parallel processing with third order is that the area occupied by the circuit is minimized as shown in Fig. 5(b). Thus, the power dissipation of the circuit is minimized by optimizing the area, frequency of operation even while the throughput remains constant.

6. Conclusion

In this paper, an architecture to optimize the power dissipation is proposed by trading off the throughput and area of the system. The third order parallel processing structure has been implemented using Verilog HDL and synthesized using Cadence® RTL Compiler and Cadence® Encounter SoC. From the results, it can be observed that a high throughput can be achieved by using the proposed third order parallel processing hardware. The third order parallel processing is validated as an optimized structure, since the first order parallel processing hardware dissipates less power with lower throughput, while the tenth order parallel processing structure dissipates more power to achieve increased throughput. The proposed structure consumes 30.80% more power than the first order parallel processing hardware while increasing the throughput by 60%. Similarly, it consumes 51.67% less power than that of the tenth order parallel processing hardware while compromising the throughput by 50%.

References

[1] W. Mao, "Modern Cryptography: theory and practice," *Prentice Hall Professional Technical Reference,* 2003.

[2] D. E. S. NIST, "FIPS PUB 46–3," *Federal Information Processing Standards Publication,* pp. 46-3, 1999.

[3] N. F. Pub, "197: Advanced encryption standard (AES)," *Federal information processing standards publication,* p. 311, 2001.

[4] A. S. Wander, N. Gura, H. Eberle, V. Gupta, & S. C. Shantz, (2005). Energy analysis of public-key cryptography for wireless sensor networks. In *Pervasive Computing and Communications, 2005. PerCom 2005. Third IEEE International Conference on* (pp. 324-328). IEEE.

[5] C. Adams, & S. Lloyd, (1999). Understanding public-key infrastructure: concepts, standards, and deployment considerations. Sams Publishing.

[6] N. Ahmad and S. R. Hasan, "Low-power compact composite field AES S-Box/Inv S-Box design in 65 nm CMOS using Novel XOR Gate," *The VLSI journal Integration,* pp. 333-344, 2013.

[7] S. Morioka and A. Satoh, "An optimized S-Box circuit architecture for low power AES design," *International Workshop on Cryptographic Hardware and Embedded Systems,* pp. 172-186, 2002.

[8] N. Sklavos and O. Koufopavlou, "Architectures and VLSI implementations of the AES-proposal Rijndael," *IEEE Transactions on Computers,* pp. 1454-1459, 2002.

[9] E. J. Swankoski, R. R. Brooks, V. Narayanan, M. Kandemir and M. J. Irwin, "A parallel architecture for secure FPGA symmetric encryption," *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International,* 2004.

[10] A. Hodjat and I. Verbauwhede, "Area-throughput trade-offs for fully pipelined 30 to 70 Gbits/s AES processors," *IEEE transactions on computers,* vol. 62, no. 3, pp. 536-547, 2013.