ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2015

AES-256 Encryption in Communication using LabVIEW

Sagar Paddhan[1], Padma Lohiya[2], Sudhir Shelke[3]

Research Scholar, Department E & TC, D. Y. Patil College of Engineering, Akurdi, Pune, India[1]

Assistant Professor, Department E & TC, D. Y. Patil College of Engineering, Akurdi, Pune, India[2]

Head R&D, JDM Design Technologies, Nagpur, India[3]

ABSTRACT: In this novel work a new scheme to enhance security of wireless sensor Network(WSN) gateway is presented. Security in WSN is a challenging task due to its limitation towards the low latency in processing speed and power. Preliminary results show the proposed scheme that does not have application dependency and have potential in integrated with any application of wireless sensor network. However, the technique used in encryption algorithm is advanced encryption standards (AES) and sleep scheduler for key management combined and operated with the same key size to provide node authentication and secure key exchange. The various techniques are studied and compared. It is found that for encryption the maximum time required after the clock is 3.339ns and throughput achieved is 4.196388Gbps. This proposed encryption scheme is implemented using LABVIEW 2013.

KEYWORDS: AES-256, LabVIEW

I. INTRODUCTION

The AES algorithm is a symmetric block cipher that can encrypt,(encipher), and decrypt, (decipher), information. Encryption converts data to an unintelligible form called ciphertext. Decryption of the cipher-text converts the data back into its original form, which is called plaintext. The AES algorithm is capable of using cryptographic keys of 128, 192, and 256 bits. Many algorithms were originally presented by researchers from twelve different nations. The Rijndael Algorithm was chosen since it had the best overall scores in security, performance, efficiency, implementation ability and flexibility. The Rijndael algorithm was developed by Joan Daemen of Proton World International and Vincent Fijmen of Katholieke University at Leuven. Cryptographic algorithms [1] are utilized for security services in various environments in which low monetary value and energy consumption are canonic requirements. Internet banking, Railway reservation, defense are examples of such technologies. Compared to software, significantly higher performance and lower power consumption can be achieved with devoted hardware. As Advanced Encryption Standard (AES) is a consistent encryption algorithm and having the evade choice in numerous applications, including the standard wireless technologies IEEE 802.11i, IEEE 802.15.4. The hardware implementation of the Rijndael algorithm can provide either high performance or low cost for specific applications. At backbone communication channels or at high traffic intensity it is not possible to lose processing speed, which drops the potency of the overall system while running cryptography algorithms in software [1]. Beside this phenomenon, a low cost and small design can be used in smart card applications, which allows a wide range of equipment to operate securely. In this research work we present an AES encryption suited for low-cost and least area requirements over communication system to provide secured wireless communication [4] The rest of the paper is organized as follows. Section II describes Rijndael AES Algorithm, Section III describes improved AES Algorithm in LabVIEW, Section IV describes Scheduler for key management , and Section V describes Simulation results. Finally the paper is concluded in Section VI.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2015

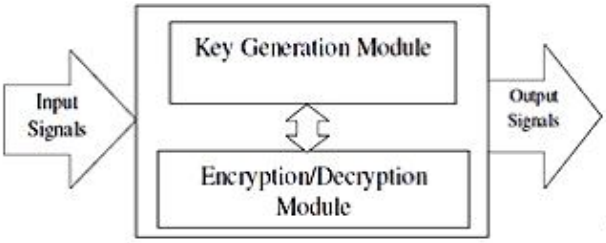

Fig. 1. Simple model for cryptographic algorithms.

II. OVERVIEW OF AES

AES is based on a design principle known as a substitution transposition network, and is fast in both software and hardware. Unlike its predecessor DES, AES does not use a Feistel network. AES is a variant of Rijndael which has a mixed block size of 128 bits, and a key size of 128, 192, or 256 bits. By contrast, the Rijndael specification per se is specified with block and key sizes that may be any multiple of 32 bits, both with a minimum of 128 and a maximum of 256 bits. AES operates on a 4 X 4 column-major order matrix of bytes, termed the state, although some versions of Rijndael have a larger block size and have additional columns in the state. Most AES calculations are done in a special finite field. Rijndael algorithm consists of encryption, decryption and key schedule algorithm It is shown as Figure 1. The main operations of the encryption algorithm among the three parts of Rijndael algorithm include [6]

A. Bytes substitution (SubBytes): Subbyte transformation is a non linear byte substitution. Each byte of the block is replaced by its substitute in an S-box i.e. each byte of state is replaced by byte in row (left 4-bits) & column (right 4-bits).S-box is constructed using a transformation of the values in GF(28). InvSubBytes are same routine as SubBytes, but uses the inverse SBox. Inverse Sbox is computed by applying the inverse affine transformation and then substituting with the multiplicative inverse, of the cells value in the SBox.

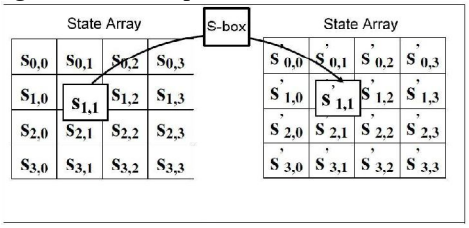

Fig. 2. Sub Byte Operation

B. Row shift (ShiftRows): ShiftRows is a simple shifting transformation. First row of the state is kept as it is, while the second, third and fourth rows cyclically shifted by one byte, two bytes and three bytes to the left, respectively. In the InvShiftRows, the first row of the State does not change, while the rest of the rows are cyclically shifted to the right by the same offset as that in the ShiftRows.

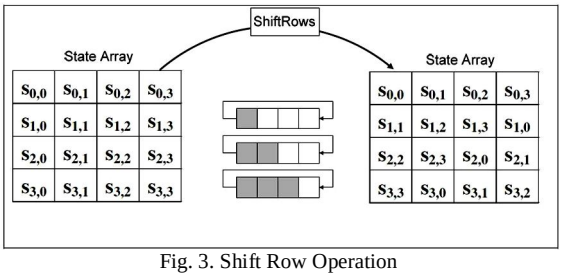

Fig. 3. Shift Row Operation

C. Column mixing (MixColumns): The MixColumns() transformation operates on the State column-by-column, treating each column as a four-term polynomial. The columns are considered as polynomials over GF(2^8) and multiplied modulo $x^4 + 1$ with a fixed polynomial a(x), given by $a(x) = (03)x^3 + (01)x^2 + (01)x + (02)$. The function xtime is used to represent the multiplication with 02, modulo the irreducible polynomial

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2015

$m(x) = x^8 + x^4 + x^3 + x + 1$. Implementation of function xtime() includes shifting and conditional xor with 1B. This transformation together with ShiftRows, provide substantial diffusion in the cipher meaning that the result of the cipher depends on the cipher inputs in a very complex way. In other words, in a cipher with a good diffusion, a single bit change in the plaintext will completely change the ciphertext in an unpredictable manner.

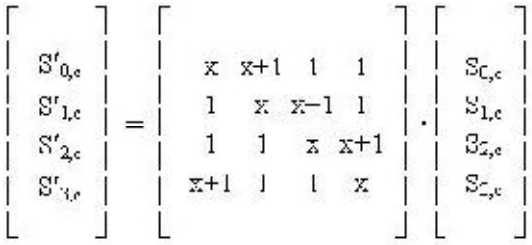

Fig. 4. Mix Coloumn Operation

D. Round key adding (AddRoundKey): Add RoundKey involves only bit-wise XOR operation. After every round output of the mixcolumn is added with round key. The round key values are added to the columns of the state in the following way. During the AddRoundKey transformation, the round key values are added to the State by means of a simple Exclusive Or (XOR) operation. Each round key consists of Nb words that are generated from the KeyExpansion routine. By inverting the encryption structure one can easily derive the decryption structure. However, the sequence of the transformations will be different from that in encryption. This feature prohibits resource sharing between encryptors and decryptors. There is no need of Inv Add RoundKey in the decryption since XOR operation is inverse of itself.

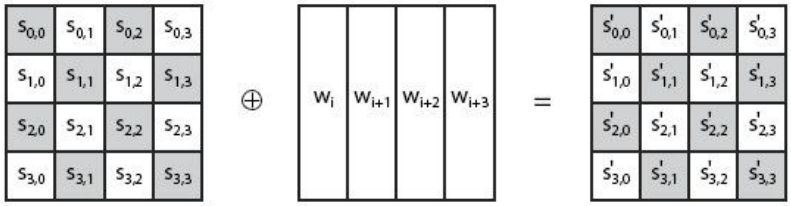

Fig. 5. Add Round Key Operation

E. Key Expansion and Key extraction: In the AES algorithm, the key expansion module is used for generating round keys for every round. There are two approaches to provide round keys. One is to pre-compute and store all the round keys, and the other one is to produce them on-the-fly. First approach consumes more area. In second approach, the initial key is divided into N_k words (key0, key1,. . . , keyNk-1) which are used as initial words. With the help of these initial words rest the words are generated iteratively. It can be computed that is 4, 6, or 8, when the key length is 128, 192 or 256-bit, respectively. Each round key has 128 bits, and is formed by concatenating four words as shown in the Fig. 6. The AES algorithm requires four words of round keys for each encryption round. That is total of $4(N_r + 1)$ round keys considering the initial set of keys required for the first AddRoundKey transformation. All the round keys are derived from the cipher key itself. This module is implemented basically the same with the traditional way as another part of the AES encryption algorithm. The only difference lies on the mode of data transmission. The initial key and expanded keys are divided into four 32-bit data before being extracted[2].

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2015

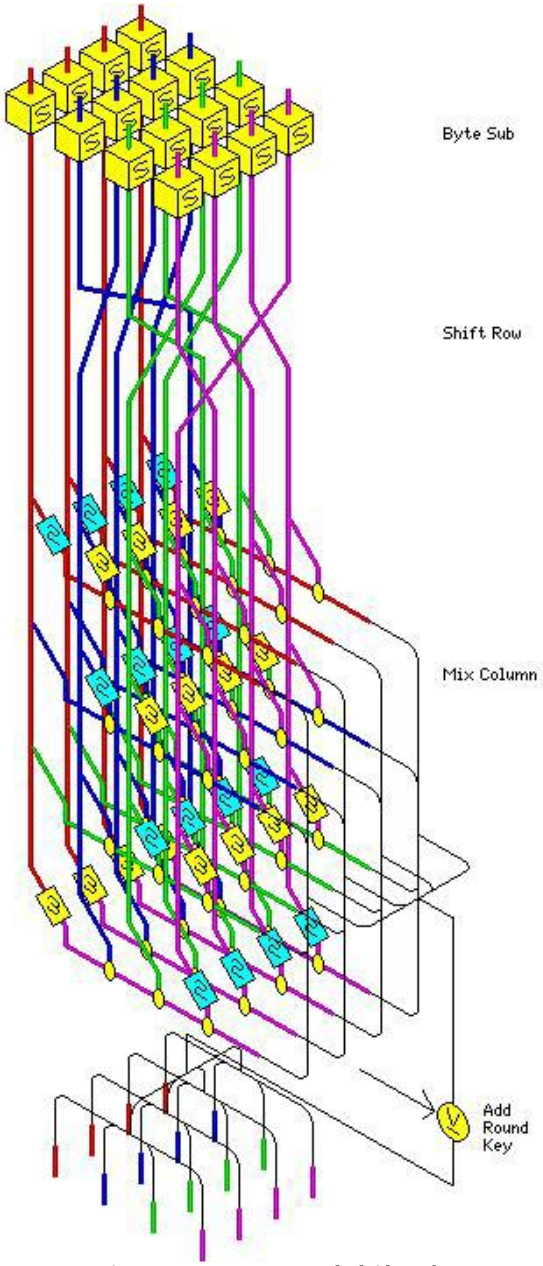

Fig. 6. Key Expansion in Rijindael Algorithm

TABLE I: NUMBER OF ROUNDS

	Key length (words)	Number of rounds (Nr)
AES 128	4	10
AES 192	6	12
AES 256	8	14

III. DESIGN METHODOLOGY WITH LABVIEW

Implementing the AES method with a device involves several steps that are described below. This is broken in different steps [14].

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2015

A. The first step in the code is the key production which occurs on the server. In this step the user will provide a secret key which is then expanded by using Rijndael key program. The short key is enlarged into a larger one. A 128-bit key is transformed into a 176-byte key, a 192-bit key is transformed into a 208-byte key, and a 256-bit key is transformed into a 240-byte key. The key schedule will take a 4-byte subset of the key as a number 32 bytes (256-bits) and an iteration count and send this data to the key schedule core, which returns a 32 bytes (256-bits).

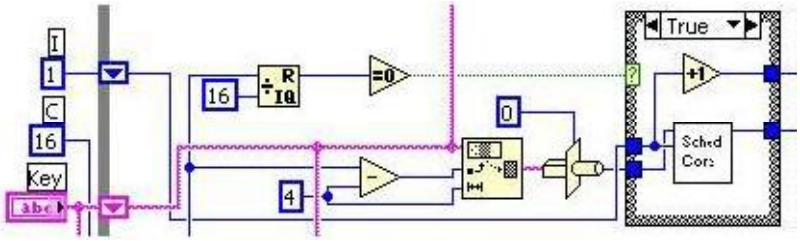

Fig. 7. Key Generation in LabVIEW

1) The key schedule core first performs a byte rotation on the key and splits the 32 bytes (256-bits) into respective bytes and sends them to have Rijndael S-box applied. This is used to apart the relationship between the key and the ciphertext [5].

2) The next step in the core scheduler is the Rcon step. This takes the first byte of the output word (from the Sbox) and performs an XOR on the byte with the result of the Rcon step, which is essentially an exponentiation of 2 to the iteration count of the key schedule [5].

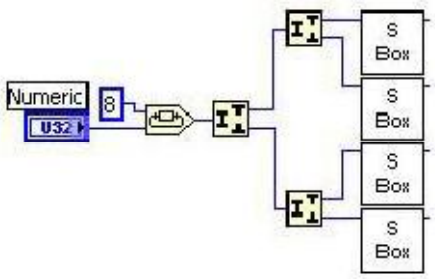

Fig. 8. S Block Implementation

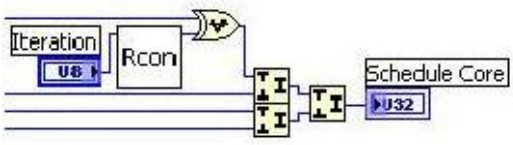

Fig. 9. Modulo Operation in S Block Implementation

B. After the key has been generated, information about the key, the type of encryption to perform, as well as the key itself, needs to be transmit to the scheduler. The key is transformed into a byte array, and sent one byte at a time along with an interrupt to ensure the scheduler receives all the data in the correct order.
C. The server will then send basic commands to the scheduler stepping the data through both the encryption and decryption process before finally returning the data back to the user.
D. In order for the server to be able to perform the cryptanalysis it is ideal to store the key in a look-up-table (LUT). For this, we use a VI methodology. As there are multiple cryptographic algorithms out there, the LUT can be used to store many different keys which will provide user adaptability.
E. Once the key information has been provided from the server, the encryption can take place as soon as the host gives the signal. The encryption will first generate what is called a round key which is derived from the key schedule. It will then begin the iterative process to perform the actual encryption.
F. The first iteration of the encryption algorithm will combine each byte of the state with the round key using a bitwise XOR. The bulk of the iterations (called rounds) will perform a non-linear substitution which replaces each byte with another according to a predefined LUT. Then each row of the state will be shifted cyclically.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2015

After which there will be a mixing operation which takes place on the columns and combined the four bytes of each column, and then the round key will be added again. The final iteration is similar to the previous 'n' rounds, without the column mixing operation.

G. Key Expansion: The key expansion routine is used to produce the roundkeys from the cipherkey. The AES standard defines the key expansion operations on four byte words. A subkey is poised of four such words. The key expansion for AES-256 is comparatively univocal:

- The first subkey corresponds to the cipher key itself.
- The following words are calculated recursively from this initial set of words using a simple XOR function.
- For the words with indices that are a multiple a special transformation is used. First, the byte ordering of first is changed by cyclic left shift, and then the SubBytes function is applied to all four bytes. In the AES standard these operations are named RotWord and SubWord respectively.

Unfortunately, the key expansion routine for other cipher key lengths introduces more irregularity to this basic process Implementations that support multiple cipher key lengths suffer excessively from this problem. Although the key expansion routine contains much less hardware than a regular AES round, the critical path of the AES system is more often located within the key expansion routine as a result of the flexibility required to implement different key lengths. The Rijndael algorithm for AES 256 has been successfully implemented using LabVIEW ver2013 is depicted in Figure 12.

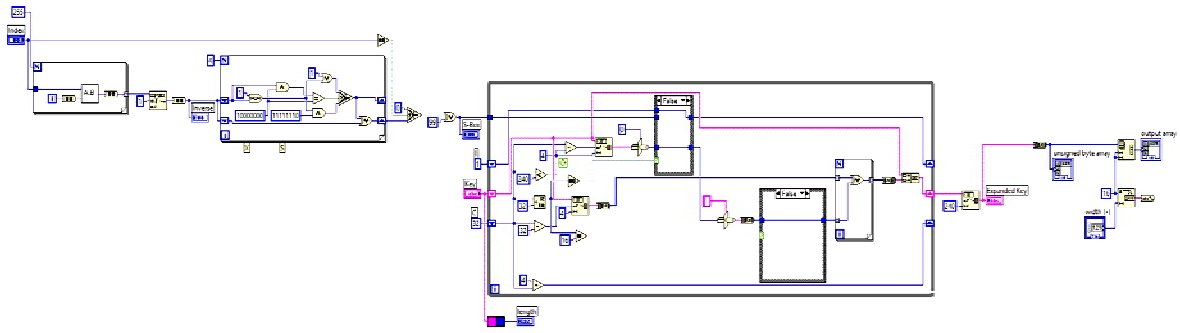

Fig. 11. Control Panel

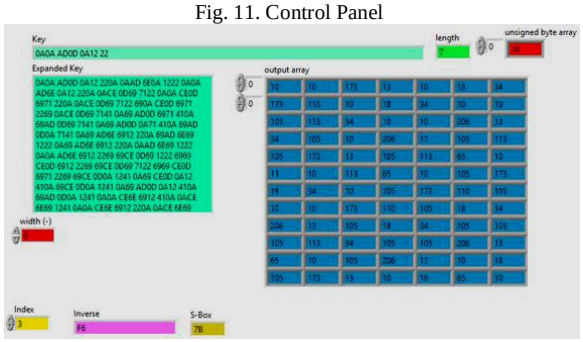

Fig. 12. Resulted output for Key Expantion

IV. POWER MANAGEMENT

We go for sleep scheduling algorithm Figure 13 for power management while transmission. Once key expand the bandwidth requirement is more at such critical stage sleep scheduling provide proper bandwidth management. Sleep scheduling protocols can be broken up into two categories: synchronous and asynchronous. Synchronous sleep scheduling policies rely on clock synchronization between round transformation in a network. Asynchronous sleep scheduling, on the other hand, does not rely on any clock synchronization between round transformation whatsoever. Nodes can send and receive packets whenever they please, according to the MAC protocol in use. shows how two nodes running asynchronous sleep schedulers a able to communicate. Nodes wake up and go to sleep periodically in the

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2015

same way they do for synchronous sleep scheduling. Since there is no time synchronization, however, there must be a way to ensure that receiving nodes are awake to hear the transmissions coming in from other nodes.

Normally preamble bytes are sent by a packet in order to synchronize the starting point of the incoming data stream between the transmitter and receiver. With asynchronous sleep scheduling, a significant number of extra preamble bytes are sent per packet in order to guarantee that a receiver has the chance to synchronize to it at some point. In the worst case, a packet will begin transmitting just as its receiver goes to sleep, and preamble bytes will have to be sent for a time equal to the receiver's sleep interval (plus a little more to allow for proper synchronization once it wakes up). Once the receiver wakes up, it synchronizes to these preamble bytes a remains on until it receives the packet. Unlike the power efficient routing protocols introduced in it doesn't make sense to have a hybrid sleep scheduling protocol based on each of the two techniques. The energy savings achieved using each of them varies from system to system and application to application. One technique is not "better" than the other in this sense, so efforts are being made to define exactly when each type should be used.

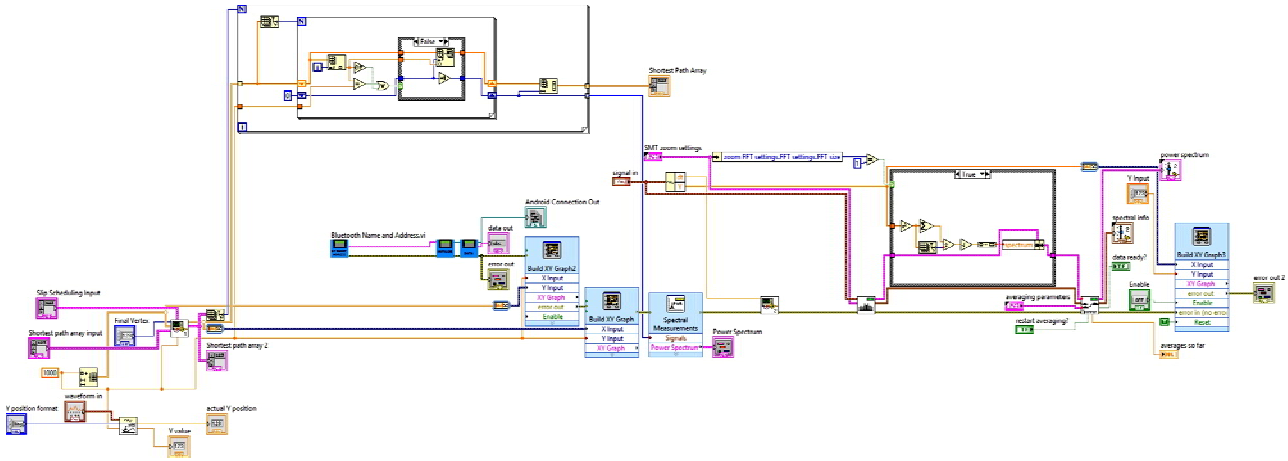

Fig. 13. Sleep Scheduling .VI Concept

V. THROUGHPUT CALCULATION

1) Maximum output required after clock (T)=3.339ns.
2) Frequency (F) = 299.49Mhz
3) Throughput for 256 AES = 4.1963Gbps.

VI. COMPARATIVE RESULT

Parameter	Work Leelavathi and Other[2]	Our Work
Maximum output time required after clock(T)	4.496ns	3.339ns
Frequency	222.41MHz	299.49MHz
Throughput	2.84Gbps	4.196Gbps

VII. CONCLUSION

The cryptographic algorithms are utilized for security services in various environments in which low cost and low power consumption are key requirements in recent. Here we deployed the service for Wireless Local Area Networks (WLAN), Wireless Personal Area Networks (WPAN), Wireless Sensor Networks (WSN), and smart cards. Compared to software, significantly higher performance and lower power consumption and maximum throughput can be achieved with dedicated hardware hence we use AES LabVIEW.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2015

REFERENCES

[1] Ai-wen luo, qing-ming yi, min shi, "Design and Implementation of Area-optimized AES Based on FPGA", IEEE 2011.
[2] Leelavathi , Prakash , Shaila , Venugopal, PAtnaik. "Design and Implementation of Advance Encryption Algorithm" . IJREAT, ISSN 2320-8791. Volume 1 , Issue 3, 2013.
[3] Ahmed Rady, Ehab EL Sehely, A.M. EL Hennawy, "Design and Implementation of area optimized AES algorithm on reconfigurable FPGA", IEEE ICM 2007.
[4] Ashwini M. Deshpande, Mangesh S. Deshpande and Devendra N. Kayatanavar, "FPGA Implementation of AES Encryption and Decryption", international conference on control, automation, communication and energy conservation -2009, 4th-6th June 2009 .
[5] Sagar Paddhan, Trupti Wagh, Sudhir N. Shelke, " Secured wireless communication using AES-256" Global Journal of Advanced Engineering and Technology, current issue, Dec-2014
[6] Sagar Paddhan, Padma Lohiya , Sudhir Shelke, "AES-256 Encryption for Secured Communication". International Journal of Emerging trends in Engineering management and research, Volume I , Issue –II, 2015.
[7] Abdel-hafeez.S.,Sawalmeh.A. and Bataineh.S., "High Performance AES Design using Pipelining Structure over GF(28)". IEEE Inter Conf.Signal Proc and Com.,vol.24-27, pp.716-719,Nov. 2007
[8] J.Yang, J.Ding, N.Li and Y.X.Guo,"FPGA-based design and implementation of reduced AES algorithm". IEEE Inter.Conf. Chal Envir Sci Com Engin(CESCE).,Vol.02, Issue.5-6, pp.67-70, Jun 2010.
[9] A.M.Deshpande, M.S.Deshpande and D.N.Kayatanavar, "FPGA Implementation of AES Encryption and Decryption" IEEE Inter.Conf.Cont,Auto,Com,and Ener., vol.01,issue04, pp.1-6,Jun.2009.
[10] Hiremath.S. and Suma.M.S., "Advanced Encryption Standard Implemented on FPGA" IEEE Inter.Conf. Comp Elec Engin.(IECEE),vol.02,issue.28,pp.656-660,Dec.2009.
[11] Rizk.M.R.M. and Morsy, M., "Optimized Area and Optimized Speed Hardware Implementations of AES on FPGA", IEEE Inter Conf. DesignTes Wor.,vol.1,issue.16,pp.207-217, Dec. 2007.
[12] Ahmed Rady, Ehab EL Sehely, A.M.EL Hennawy, "Design and Implementation of area optimized AES algorithm on reconfigurableFPGA",IEEEInter.conf.CompElecEngin(IECEE),978-1-4244-1847-3/07.2007.
[13] S.Sankar Ganesh,J.Jean Jenifer Nesam, "FPGA Based SCA Resistant AES S-BOX Design", International Journal of Scientific & Engineering Research,volume4,Issue 4,pp.1143-1149,April-2013.
[14] www.ni.com
[15] NIST, Advanced Encryption Standard (AES), NIST, FIPS-197, 2001.