Department of Computer Science and Engineering (CSE)
Islamic University of Technology

Lightweight AES Algorithm for Resource Constraint Devices

Authors
Arnab Rahman Chowdhury - 124418
and
Junayed Mahmud - 124421

Supervisor
Dr. Abu Raihan Mostofa Kamal
Associate Professor
Department of CSE
Islamic University of Technology

A thesis submitted to the Department of CSE
in partial fulfillment of the requiremnts for the degree of B.Sc.
Engineering in CSE
Academic Year: 2015-16

November - 2016

Declaration of Authorship

This is to certify that the work presented in this thesis is the outcome of the analysis and experiments carried out by Arnab Rahman Chowdhury and Junayed Mahmud under the supervision of Dr. Abu Raihan Mostofa Kamal, Associate Professor, Department of Computer Science and Engineering (CSE), Islamic University of Technology (IUT), Dhaka, Bangladesh. It is also declared that neither of this thesis nor any part of this thesis has been submitted anywhere else for any degree or diploma. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

Authors:

Arnab Rahman Chowdhury
Student ID - 124418

Junayed Mahmud
Student ID - 124421

Supervisor:

Dr. Abu Raihan Mostofa Kamal
Associate Professor
Department of CSE
Islamic University of Technology

Acknowledgement

We would like to express our grateful appreciation for **Dr. Abu Raihan Mostofa Kamal**, Associate Professor, Department of Computer Science & Engineering, IUT for being our advisor and mentor. His motivation, suggestions and insights for this thesis have been invaluable. Without his support and proper guidance this research would not have been possible. His valuable opinion, time and input provided throughout the thesis work, from first phase of thesis topics introduction, subject selection, proposing algorithm, modification till the project implementation and finalization which helped us to do our thesis work in proper way. We are really grateful to him.

We are also grateful to **Mr. Tariq Tonmoy**, M.Sc student & Lecturer, Department of Computer Science & Engineering, Islamic University of Technology for his valuable suggestions on our proposal of biomedical content based image retrieval.

Abstract

Our world is slowly approaching brink of mankind's next technological revolution since the conception of Internet of Things(IoT) emerges. The realization of IoT requires an enormous amount of sensor nodes which have limited battery power, memory, compu tational latency and communication bandwidth to acquire inputs from the connected objects. In the current developments of the resource constraint environments, the trend is shifted towards lightweight cryptographic algorithm. Many lightweight cryptograpic algorithms have been developed and also existing algorithms are modied in terms of resource constraint environment. In this paper we analyzed a very popular block cipher AES-128 and tried to make it lightweight regarding energy consumption. We also studied some other lightweight block cipher which are optimized in terms of energy, power and memory.

Contents

1 Introduction

1.1 Overview

A dramatical change and development has been occurred since the buzzword "Internet" has been introduced by Sir Tim Berners-Lee on march 12,1989. Ever since then, the Internet has been integrated into ours day to day lives as an essential need at an incredible pace.The geographical barriers between nations has destroyed and the concept of "globalization" has emerged. More importantly, it became the besis of mankind's technological advancement.

Internet of Things(IoT) is the next revolution of Internet which brings such an impact on our everyday lives. IoT is the extension of the Internet to connect just about everything on the planet. This includes real and physical objects ranging from household accessories to health monitoring sensors to industrial controls. As such, these "things" that are connected to the Internet will be able to take actions or make decisions based on the information they obtained from the Internet with or without human intervention. In addition, they also update the Internet with real time information with the help of various sensors. This allows objects connected to the IoT to communicate and work with each other seamlessly.

IoT works with resource constraint components. Those components have limited amount of battery power, memory. Also those component communicate through the Internet which is not secured. As mentioned previously, IoT relies heavily on the sensor nodes on the connected objects to obtain real time information. In certain applications, condentiality of such information might be extremely important. Therefore, cryptography comes into play.

Due to the resource constrained nature of the sensors, the security requirement has led to various attempts to optimize established cryptographic ciphers for small scale applications. Marginal amount of security is needed for lightweight device like sensors.The most notifying aspect of lightweight device is it's computational speed, battery power etc. Energy consumption is an important issue regarding lightweight devices. Many standard cryptographic algorithm has been introduced which is best in terms of security but not in energy consumption. Two main approaches can be followed to design and implement security primitives tting the needs of extremely constrained devices: designing new algorithms to be implemented into constrained devices, or trying to implement standards and known algorithms in a lightweight fashion, eventually relaxing the per formance constraints. Some recently proposed lightweight cryptography are: Present[1], Prince[2], Simon/Speck[3] etc. Possible example of the second approach are implementations of the Advanced Encryption Standard algorithm (AES)[4], SHA-256[5] etc.

In this paper, we provide a hardware implementation of Standalone AES Encryption in CC2420(TinyOS 2.1.2 and telosb). We analyze the energy consumption result by sending temperature value which is sensed by telosb from one mote to other with and without encryption. Analyzing many related works we came to know that Substitution Box(S-Box) and Mix Column are the most energy consuming parts of AES. We focused on S-Box and modified it to make it energy efficient. We worked in Galois Field 2^4 instead of 2^8 in our proposed work. We implemented our proposed modified AES in

both C language and TinyOS and compared energy consumption with the original one. In therms of memory complexity our proposed AES is better then the original AES. Analyzing the packet vs voltage graph we can conclude that our proposed AES is energy efficient then the Rijndael AES.

1.2 Problem Statement

Internet of Things(IoT) devices are lightweight and resource constraint. Those components have limited amount of battery power, memory. Security issue comes when IoT device communicate with each other. Dierent cryptographic algorithm are used to secure the data transmission. But due to resource constraint issue, all standard encryption decryption algorithms can't be used. Many lightweight cryptographic algorithms have already proposed which provides good result regarding energy consumption. We are proposing a light version of standard block cipher AES-128 for IoT devices.

1.3 Motivation & Scopes

If we classify cryptographic algorithms, we will find 2 major classifications.

- Symmetric Crypto System
- Asymmetric Crypto System

A secret key algorithm (sometimes called a symmetric algorithm) is a cryptographic algorithm that uses the same key to encrypt and decrypt data. The keys represent a shared secret between two or more parties that can be used to maintain a private information link. The main drawback of Symmetric Crypto System is that both parties have access to the secret key. Symmetric-key encryption can use either stream ciphers or block ciphers. Some important block ciphers are AES, Blowfish DES (Internal Mechanics, Triple DES) etc. Widely used stream ciphers are RC4, Block ciphers in stream mode, ChaCha etc.

Public key cryptography, or asymmetric cryptography, is any cryptographic system that uses pairs of keys: public keys which may be disseminated widely, and private keys which are known only to the owner. This accomplishes two functions: authentication, which is when the public key is used to verify that a holder of the paired private key sent the message, and encryption, whereby only the holder of the paired private key can decrypt the message encrypted with the public key. Some well regarded Asymmetric Crypto Systems are Diffie–Hellman key exchange protocol, DSS (Digital Signature Standard), Elliptic Curve Crypto System, RSA encryption algorithm etc.

Based on many related research we come to know that Asymmetric Cryptography performs complex mathematical calculations which actually drains the battery power very rapidly. In terms of security purpose they are well efficient and demanding but for resource constraint devices (IoT devices), they are not best suited. Moreover, moderate security is needed for IoT devices as they work in real-time environment and have to send data continuously. So, We choose symmetric block cipher which actually performs less computation than asymmetric crypto system and also protect the data from intrusion through providing considerable security.

Some of the mojor motivation and scopes are enlisted belows:

- Resource constraint components have limited amount of battery power, memory.
- Optimization of AES-128 as lightweight block cipher.
- Secure data transmission between lightweight devices.

1.4 Research Challenges

We are working with AES-128 algorithm which is used in resource constraint devices. Though it is energy efficient in comparison to other algorithms, yet we need to face lots of challenges which modifying the algorithm. The core challenges are enlisted below:

- Huge computational power due to number of rounds according to key size
- Energy consumption issue
- S-box and Mix-column operation

1.5 Thesis Outline

In Chapter 1 we have discussed our study in a precise and concise manner. Chapter 2 deals with the necessary literature review for our study and their development so far. In Chapter 3 we have described the original Methodology of AES. in Chapter 4 we have discussed about the original S-Box generation method and in following chapter 5 we have stated our proposed methodology of generating S-Box. In chapter 6 we have discussed about our analyzed result with the final segment of this study which contains all the references and credits used.

2 Literature Review

Regarding Resource constraint environment, many renowned research works have been done in recent years. Many lightweight cryptographic algorithms have been proposed targeting low area and low power. Some standard block ciphers are also modified for resource constraint environment. Despite a large number of previous works targeting area and power, only limited were devoted to the optimization of the energy parameter. Energy and power are correlated parameters. Power is the amount of energy consumed per unit time. The time integral of power is measured as energy consumption.

2.0.1 Energy Effciency of Lightweight Block Cipher

Focusing on block ciphers in particular, it is important to notice that AES still remains the preferred choice for providing security also in constrained devices, even if some lightweight algorithms are now standardized. Several implementations of AES and its basic transformations (such as S-boxes) targeting low area and low power were proposed in the past, for example, the implementation of Feldhofer et al.[6] and the one of Moradi et al.[7]. The first design is based on a 8-bit datapath, and occupies approximately 3400 Gate Equivalents (GE). The second design features a mixed data path and requires approximately 2400 GE. Both researches are hardware based. Hocquet et al.[8] showed that by exploiting technological advances and algorithmic optimization the AES core, energy can be consumed as little as 740 pJ per encryption. This implementation is also hardware based.

The most significant previous works on this area are the one of Kerckhof et al.[9] and the one of Batina et al.[10]. The first work, addresses the problem of efficiency for lightweight designs. The authors present a comprehensive study comparing a number of algorithms using different metrics such as area, throughput, power, and energy, and applying state of the art techniques for reducing power consumption such as voltage scaling. However, the evaluation reported in the paper is at very high level and concentrates only on a specific implementation, without considering the effects on energy consumption of different design choices, such as size of the datapath, amount of serialization, or effects of architectural optimization applied at each stage of the algorithm. The second work explores area, power, and energy consumption of several recently-developed lightweight block ciphers and compares it with the AES algorithm, considering also possible optimization for the non linear transformation. However, no possible optimization was considered for other transformations, and effects of other design choices, such as serialization were not considered in the work.

Bogdanov et. al.[11] have investigated how the variation in (a) architecture of SBox and MixCloumn, (b) frequency of the clock cycle and (c) unrolling the design can affect the energy consumption. They have analyzed the energy consumption figures of several lightweight ciphers(AES-128, Present, Piccolo, TWINE etc.) with different degrees of unrolling. They have proved that the total energy consumed in a circuit during an encryption operation has roughly a quadratic relation with he degree of unrolling. They have also analyzed that the substitution layer consumes the most part of the energy in the round based design and the 2-round unrolled designs respectively.Again, They have found that in the 2-round unrolled design, the second round functions consume more energy than the first due to switching activity.

2.0.2 Energy Consumption of Lightweight Blockciphers in FPGA

Banik et. al.[12] have tackled the problem of energy effciency of lightweight blockciphers implemented in re-configurable devices(Field Programmable Gate Array) for the first time and explored the effects that round unrolling might have on energy consumption. Their result have shown that Present is the most energy efficient algorithm and 2-round unrolling design is the best one. In this paper, authors have compared energy consumption of 7 different lightweight blockciphers which are implemented in re- configurable devices(FPGA).

Wong et. al.[13] have implemented a compact PRESENT cipher on a FPGA platform and proposed a design uses a 8-bit datapath by implementing and factorizing a Boolean S-Box through Karnaugh mapping instead of traditional look-up-table which gives the smallest FPGA implementation of the PRESENT cipher to date, requiring only 62 slices. Though the result of their proposal did not produce the highest throughput but the main competing factor falls on the area requirement of the designs and they achieved the best results, requiring only 62 slices. though this paper is not related to energy consumption on RCE but implementation may help us to develop a better design in terms of energy consumption.

2.0.3 Survey of Modern Symmetric Cryptographic Solutions for RCE

To encourage the researchers, Kong et.al.[14] have surveyed modern symmetric cryptgraphic solutions for resource constrained environments and 100 selected symmetric ciphers(38 block ciphers, 6 involution ciphers, 28 lightweight block ciphers and 28 stream ciphers). Authors have provided collective surveys from other litterateurs on the topic of hardware, design requirements, and security trends of RCEs. Selection of a suitable cryptographic algorithm for an application or an environment, the understandings of both the algorithmic requirements in terms of hardware and the specifications of the development platform intended has to be established. In this paper, specification of the symmetric block ciphers are provided with application and design environment. Most of the implementation of the selected ciphers are on re-configurable devices (FPGA) and the results are analyzed in terms of Gate Equivalent (GE). Basically hardware implementation results are focused to survey different types of block ciphers.

2.0.4 Energy Efficient Security Architecture for WSN

Since we are working with telosb sensor nodes which works in Wireless Network, we have gone through some paper related to energy consumption in Wireless Sensor Networks. Zivkovic et. al.[15] have worked with energy efficient security architecture. Basically we are working on symmetric cryptographic algothims (e.g. AES-128) but this paper is based on public key cryptography methods. From many analysis we can see that public-key cryptography methods have been long considered as energy inefficient to be applied in constraint environments such as sensor networks. This paper is a first step in designing a java security framework for wireless sensor networks whose primary goal is to be energy efficient. For this purpose, they have proposed a heterogeneous WSN consisting of two java enabled cluster head and numerous low power sensor nodes which are clustered together. Low-power sensor nodes senses data from the environment and cluster head performs powerful data processing. Node clustering is done due to limit the number of communication which consume a lot of energy. The security architecture

implies the use of elliptic curve Diffie Hellman for key exchange and hardware enabled AES encryption for secure communication.

2.0.5 Modified AES Algorithm Using Multiple S-Boxes

Felicisimo et.al.[16] propose to implement two substitution boxes, where the first Sbox is the Rijndael Sbox. The second Sbox was constructed through an XOR operation and affine transformation and will replace the MixColumns operation within the internal rounds in the cipher. Based on simulation testing conducted, it was found out that the modified AES algorithm using multiple SBoxes has better speed performance compared to the original cipher. Authors used three steps to generate substitution box instead of MixColumn operation. They are given below:

- **Exclusive OR Operation :** XOR using some key[i] which is a hexadecimal value ranging from 0x00 to 0xFF.
- **Affine transformation :** To scramble the bits in each byte value, they next apply the following transformation to each bit.

$$b_i = b_i \; XOR \; b_{(i+4) \; mod \; 8} XOR \; b_{(i+5) \; mod \; 8} XOR \; b_{(i+6) \; mod \; 8} XOR \; b_{(i+7) \; mod \; 8} \; XOR \; C_i$$

$$b_i = i^{th} \; bit \; of \; input \; byte$$

$$C_i = i^{th} \; bit \; of \; a \; specially \; designated \; byte \; which \; is \; 0x63$$

- **Matrix Mapping Operation :** Map each value to the matrix as appropriate to create the final look-up tables.

The final AES2SboxXOR$_{7F}$ and final inverse AES2SboxXOR$_{7F}$ are shown in Fig 1 and Fig 2 respectively.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	13	0C	2D	32	6F	70	51	4E	EB	F4	D5	CA	97	88	A9	B6
1	E2	FD	DC	C3	9E	81	A0	BE	1A	05	24	3B	66	79	58	47
2	F0	EF	CE	D1	8C	93	B2	AD	08	17	36	29	74	6B	4A	55
3	01	1E	3F	20	7D	62	43	5C	F9	E6	C7	D8	85	9A	BB	A4
4	D4	CB	EA	F5	A8	B7	96	89	2C	33	12	0D	50	4F	6E	71
5	25	3A	1B	04	59	46	67	78	DD	C2	E3	FC	A1	BE	9F	80
6	37	28	09	16	4B	54	75	6A	CF	D0	F1	EE	B3	AC	8D	92
7	C6	D9	F8	E7	BA	A5	84	9B	3E	21	00	1F	42	5D	7C	63
8	9C	83	A2	BD	E0	FF	DE	C1	64	7B	5A	45	18	07	26	39
9	6D	72	53	4C	11	0E	2F	30	95	8A	AB	B4	E9	F6	D7	C8
A	7F	60	41	5E	03	1C	3D	22	87	98	B9	A6	FB	E4	C5	DA
B	8E	91	B0	AF	F2	ED	CC	D3	76	69	48	57	0A	15	34	2B
C	5B	44	65	7A	27	38	19	06	A3	BC	9D	82	DF	C0	E1	FE
D	AA	B5	94	8B	D6	C9	E8	F7	52	4D	6C	73	2E	31	10	0F
E	B8	A7	86	99	C4	DB	FA	E5	40	5F	7E	61	3C	23	02	1D
F	49	56	77	68	35	2A	0B	14	B1	AE	8F	90	CD	D2	F3	EC

Figure 1: AES2SboxXOR$_{7F}$

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	7A	30	EE	A4	53	19	C7	8D	28	62	BC	F6	01	4B	95	DF
1	DE	94	4A	00	F7	BD	63	29	8C	C6	18	52	A5	EF	31	7B
2	33	79	A7	ED	1A	50	8E	C4	61	2B	F5	BF	48	02	DC	96
3	97	DD	03	49	BE	F4	2A	60	C5	8F	51	1B	EC	A6	78	32
4	E8	A2	7C	36	C1	8B	55	1F	BA	F0	2E	64	93	D9	07	4D
5	4C	06	D8	92	65	2F	F1	BB	1E	54	8A	C0	37	7D	A3	E9
6	A1	EB	35	7F	88	C2	1C	56	F3	B9	67	2D	DA	90	4E	04
7	05	4F	91	DB	2C	66	B8	F2	57	1D	C3	89	7E	34	EA	A0
8	5F	15	CB	81	76	3C	E2	A8	0D	47	99	D3	24	6E	B0	FA
9	FB	B1	6F	25	D2	98	46	0C	A9	E3	3D	77	80	CA	14	5E
A	16	5C	82	C8	3F	75	AB	E1	44	0E	D0	9A	6D	27	F9	B3
B	B2	F8	26	6C	9B	D1	0F	45	E0	AA	74	3E	C9	83	5D	17
C	CD	87	59	13	E4	AE	70	3A	9F	D5	0B	41	B6	FC	22	68
D	69	23	FD	B7	40	0A	D4	9E	3B	71	AF	E5	12	58	86	CC
E	84	CE	10	5A	AD	E7	39	73	D6	9C	42	08	FF	B5	6B	21
F	20	6A	B4	FE	09	43	9D	D7	72	38	E6	AC	5B	11	CF	85

Figure 2: AES2SboxXOR$_{7F}$

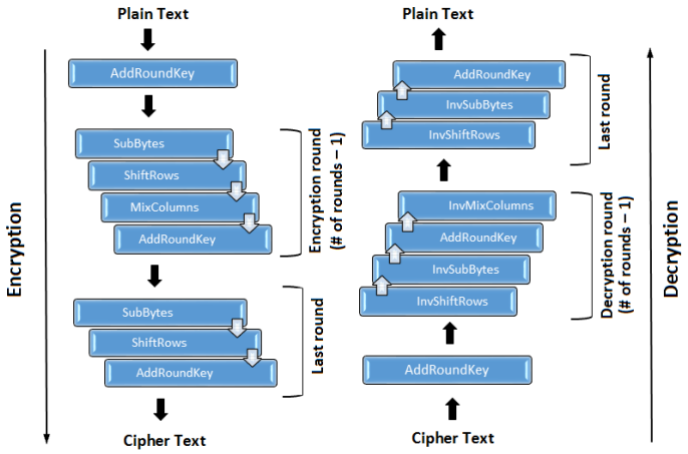

Figure 3: General design of AES encryption and decryption

3 Rijndael AES algorithm

The Advanced Encryption Standard(AES) is a symmetrickey block published by the National Institute of Standards and Technology(NIST) in December 2001. For selecting AES, three areas were defined by NIST such as security, cost, implementation and Rijndael was judged the best at meeting the combination of these criteria.

AES is a non-Feistel cipher that encrypts and decrypts a data block of 128-bits. It uses 10,12,14 rounds. The are three different key lengths. The encryption/decryption is consists of 10 rounds of processing for a 128-bit keys, 12 rounds for 192-bit keys, and 14 rounds for 256-bit keys.

AES uses several rounds in which each round is made of several stages. Data block is transformed from one stage to another. At the beginning and end of the cipher, AES uses the term data block. Before and after each stage, the data block is referred to as a state.Each round, except the last, uses four transformations that are invertible. The last round has only three transformations. Figure 3 shows the AES cipher structure.

Stages of each round are:

3.0.1 Substitute Bytes

The first transformation, SubBytes, is used at the encryption site. It's a non-linear byte substitution that operates independently on each byte of the state using a substitution table(S-Box). All the 16 bytes of the state are substituted by the corresponding values found from lookup table. In decryption, InvSubBytes, is used. States bytes are substituted from InvSubBytes table. Figure 4 shows the SubByte operation.

3.0.2 Shift Rows

In the encryption, the state bytes are shifted left in each row. It is called ShiftRows operation. The number of the shifts depends on the row number (0,1,2 or 3) of the state matrix. the row 0 bytes are not shifted and row 1, 2, 3 are shifted 1, 2, 3 bytes left accordingly. Figure 5 shows the ShiftRows operation.

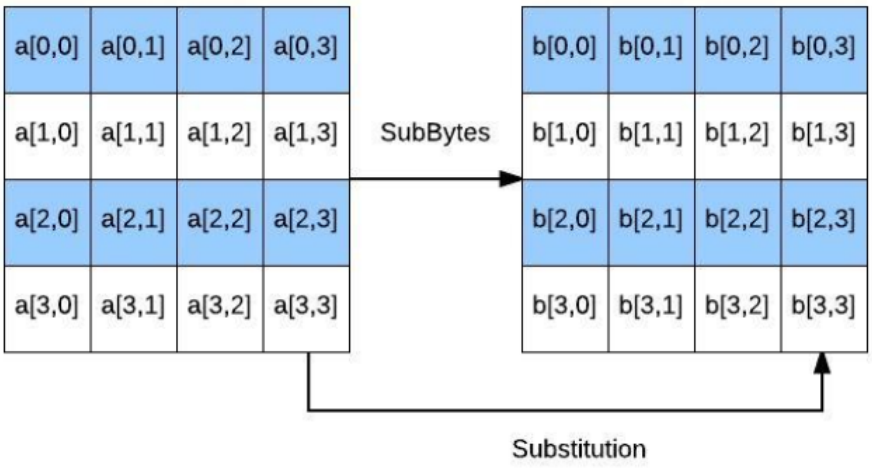

Figure 4: Byte Substitution

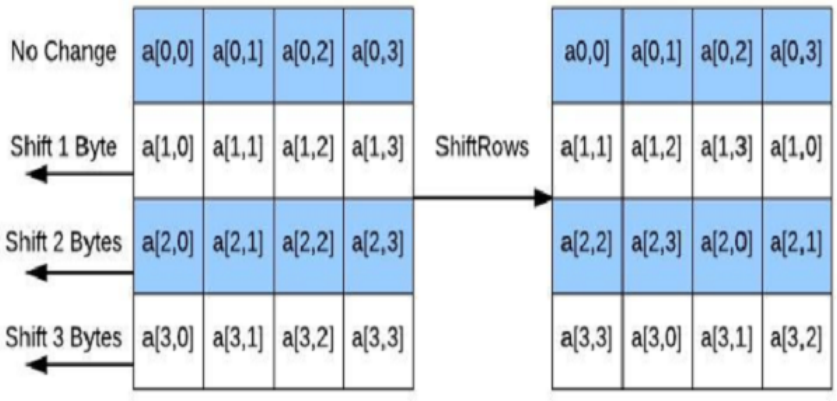

Figure 5: Shifting Rows

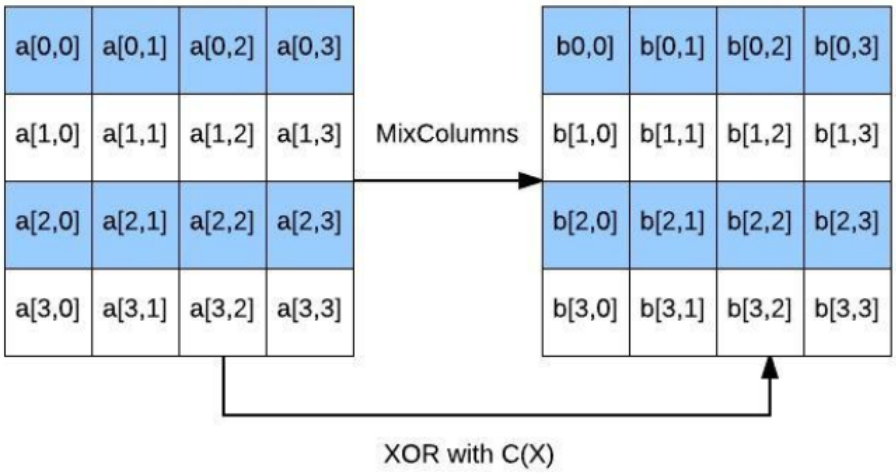

Figure 6: Mixing Columns

3.0.3 Mix Column

The MixColumn transformation operates at the column level. It transforms each column of the state to a new column. The transformation is a actually the matrix multiplication of a state column by a constant square matrix. The Galois Field is used in this operation. The bytes are treated as polynomials rather than numbers. Figure 6 shows the MixColumn operation.

3.0.4 Add Round Key

AddRoundKey proceeds one column at a time. It is similar to MixColumns in this respectr. AddRoundKey adds a round key word with each column matrix. The operation in AddRoundKey is matrix addition. Figure 7 shows the AddRoundKey operation.

In encryption, SubBytes, ShiftRows, MixColumn and AddRoundKey are performed in all rounds except the last round. MixColumn ransformation operation is not performed in the last round of encryption. The decryption process is essentially the same structure as the encryption, following the nine rounds of Inverse ShiftRows, Inverse SubBytes, Inverse AddRoundKey and Inverse MixColumns transformation. In the final round, the Inverse MixColumns is no longer performed.

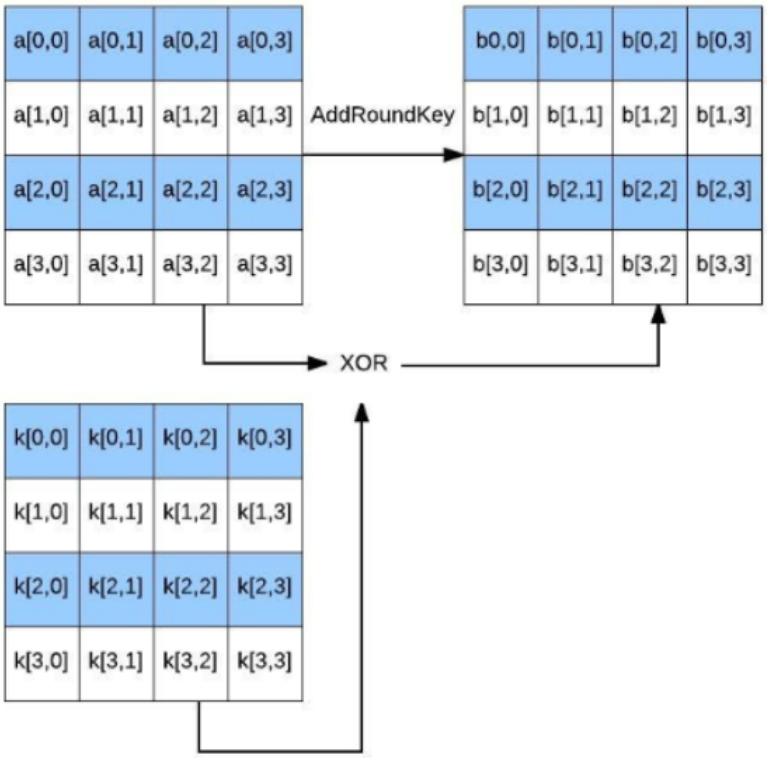

Figure 7: Adding Round Key

4 Rijndael S-Box generation method

In the original AES, S-box is generated in 2 phases.

- Multiplicative inverse phase
- Affine transformation phase

Multiplicative inverse phase : In multiplicative inverse phase, the input byte is in-versed by substituting value from Multiplicative inverse table. The multiplicative inverse table is given in Figure 8.

Affine transformation : Affine transformation consists of two operation. First, 8x8 square matrix's multiplication and secondly, 8x1 constant column matrix addition. In original AES, 0x63 is chosen as the constant column value.

The original AES S-box generation procedure is given step by step below:

- $x^8 + x^4 + x^3 + x + 1$ is used as irreducible polynomial
- b[0]...b[7] is the multiplicative inverse of input
- Affine transformation

	Y															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	00	01	8D	F6	CB	52	7B	D1	E8	4F	29	C0	B0	E1	E5	C7
1	74	B4	AA	4B	99	2B	60	5F	58	3F	FD	CC	FF	40	EE	B2
2	3A	6E	5A	F1	55	4D	A8	C9	C1	0A	98	15	30	44	A2	C2
3	2C	45	92	6C	F3	39	66	42	F2	35	20	6F	77	BB	59	19
4	1D	FE	37	67	2D	31	F5	69	A7	64	AB	13	54	25	E9	09
5	ED	5C	05	CA	4C	24	87	BF	18	3E	22	F0	51	EC	61	17
6	16	5E	AF	D3	49	A6	36	43	F4	47	91	DF	33	93	21	3B
7	79	B7	97	85	10	B5	BA	3C	B6	70	D0	06	A1	FA	81	82
X 8	83	7E	7F	80	96	73	BE	56	9B	9E	95	D9	F7	02	B9	A4
9	DE	6A	32	6D	D8	8A	84	72	2A	14	9F	88	F9	DC	89	9A
A	FB	7C	2E	C3	8F	B8	65	48	26	C8	12	4A	CE	E7	D2	62
B	0C	E0	1F	EF	11	75	78	71	A5	8E	76	3D	BD	BC	86	57
C	0B	28	2F	A3	DA	D4	E4	0F	A9	27	53	04	1B	FC	AC	E6
D	7A	07	AE	63	C5	DB	E2	EA	94	8B	C4	D5	9D	F8	90	6B
E	B1	0D	D6	EB	C6	0E	CF	AD	08	4E	D7	E3	5D	50	1E	B3
F	5B	23	38	34	68	46	03	8C	DD	9C	7D	A0	CD	1A	41	1C

Figure 8: Multiplicative inverse table

- The 8x8 matrix is formed using a formula which is given below:

$$b_i = b_i \; XOR \; b_{(i+4) \; mod \; 8} XOR \; b_{(i+5) \; mod \; 8} XOR \; b_{(i+6) \; mod \; 8} XOR \; b_{(i+7) \; mod \; 8} \; XOR \; C_i$$

$$b_i = i^{th} \; bit \; of \; input \; byte$$

$$C_i = i^{th} \; bit \; of \; a \; specially \; designated \; byte \; which \; is \; 0x63$$

- Finally we get the output byte d[0]. . . d[7]

Figure 9 shows the overall procedure from top to bottom

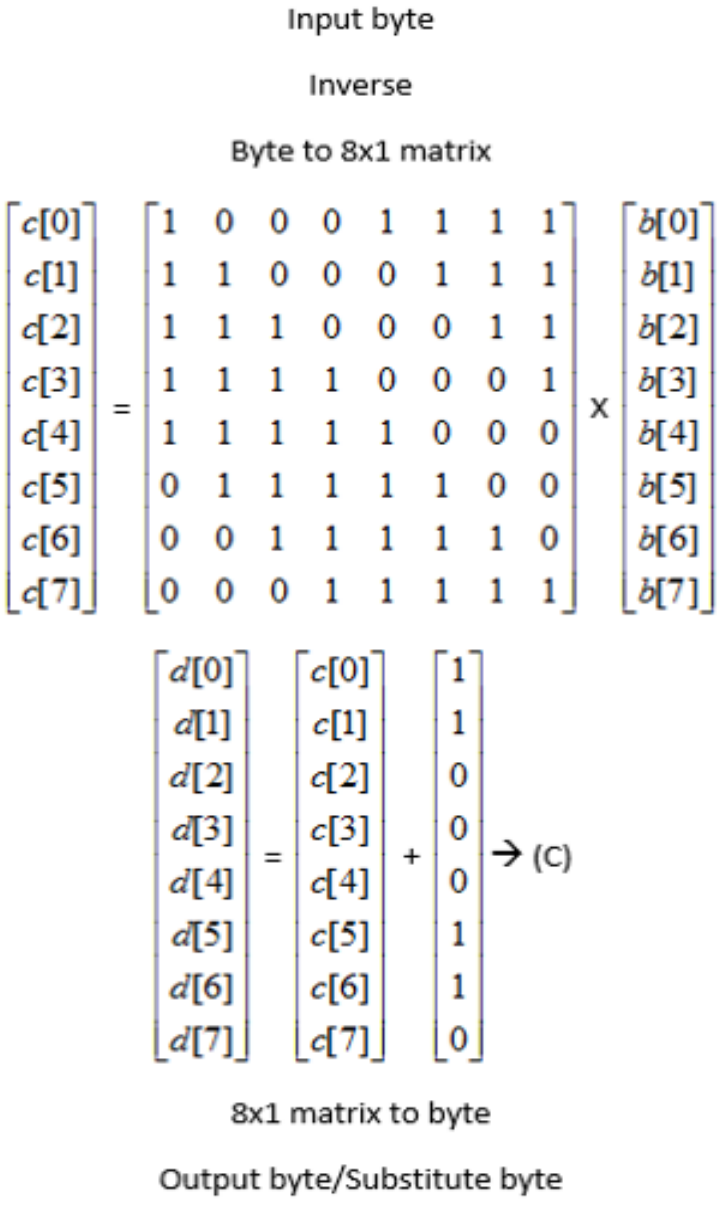

Figure 9: Original S-Box generation process

5 Proposed Method

5.1 Background Study

As cryptography term comes new to us, we have go through some basic study related to cryptography.

- Finite field or Galois Field
- Encryption and Decryption process of Advanced Encryption Standard (AES)
- Encryption and Decryption process of Data Encryption Standard (DES)
- Asymmetric cryptographic algorithm RSA
- Diffie Hellman key exchange
- Elliptic Curve Cryptography (ECC)

5.2 Hardware Implementation of Standalone AES Encryption in CC2420(TinyOS 2.1.2 and telosb)

5.2.1 TinyOS 2.1.2

TinyOS is an open source, BSD-licensed operating system designed for low-power wireless devices, such as those used in sensor networks, ubiquitous computing, personal area networks, smart buildings, and smart meters.

5.2.2 Telosb

Telosb is a low Power Wireless Sensor Module.

- TelosB is a mote from Memsic (old Crossbow) technology.
- It is composed of the MSP430 (the MSP430F1611) microcontroller.
- It has a radio chip CC2420.
- The microcontroller of this mote operates at 4.15 MHz.
- It has a 10 kBytes internal RAM and 48 KBytes of programmable memory.

Applications

- Platform for research
- Wireless sensor network experimentation

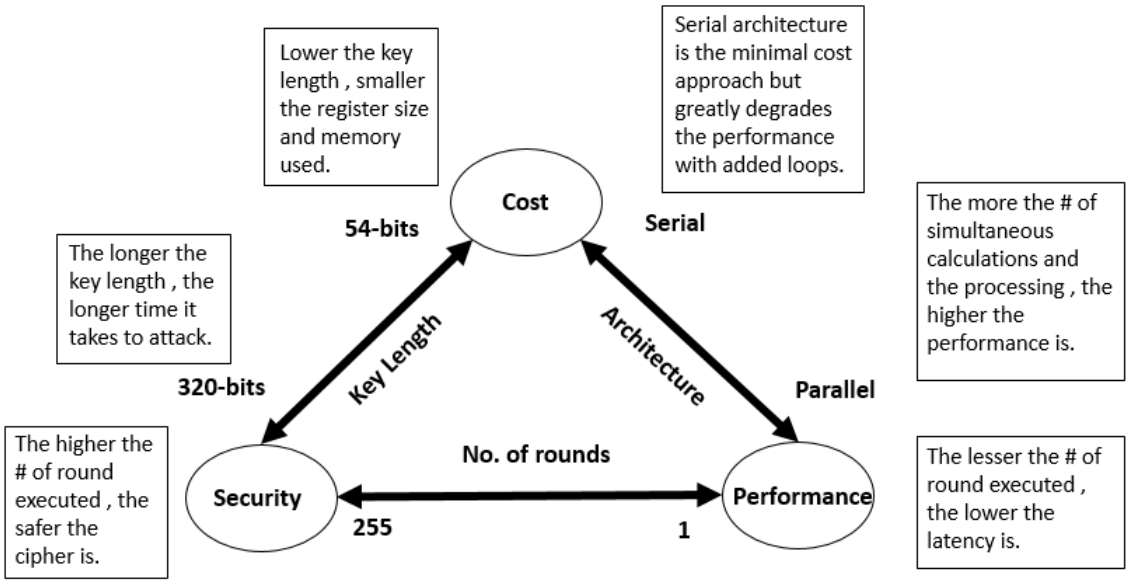

Figure 10: 3 Parameters for selecting efficient cryptographic algorithms

5.2.3 Three important parameters

Choosing appropriate Crypto System for any environment is very important. Proper selection of cryptographic algorithm can help with respect to security issue as well as energy and power. Actually three important parameters help the cryptographer for selecting appropriate algorithm for particular environment. Those parameters are

- Cost
- Security
- Performance

Figure 10 shows the relation among those three parameters and how appropriate algorithm can be chosen.

Remainder	Quotient	Auxiliary
$X^4 + X + 1$		0
$X^3 + X^2 + X + 1$		1
X	$X + 1$	$X + 1$
1	$X^2 + X + 1$	X^3

Figure 11: Extended Euclidean Algorithm

5.3 Modified AES S-Box generation

5.3.1 Multiplicative inverse table generation

Multiplicative inverse table generation is as same as the original one but we work in Galois Field(2^4) instead of Galois Field (2^8) which is used in the original S-Box generation. So, we generate a 1D Multiplicative inverse table in place of 2D Multiplicative inverse table. There are many irreducible polynomial in GF(2^4) and we choose x^4+x+1 as our irreducible polynomial. One simple example is given below to clarify the whole multiplicative inverse table generation.

First we need to find a polynomial f(x) of degree 4 which is irreducible over GF(2). There are many of these to choose from. We choose f(x) = $x^4 + x + 1$.

"Addition" is the usual addition of polynomials (reducing coefficients modulo 2), and "multiplication" is the usual multiplication of polynomials (reducing coefficients modulo 2), followed by a reduction modulo f(x) (and further coefficient reduction modulo 2) until the result has degree less than 4. Using these operations, this set forms a field isomorphic to GF(2^4).

The multiplicative inverse table generation is done by Extended Euclidean Algorithm. Suppose we want to generate inverse of 0x0F. The polynomial function of 0x0F is a(x) = $x^3 + x^2 + x + 1$.Figure 11 shows the process of inversion.

Steps :

- The Auxiliary column always starts with 0 and 1. The Remainder column always starts with f(x) and a(x).
- To fill in any subsequent row, divide the remainders in the previous two rows, and put the quotient in the Quotient column and the remainder in the Remainder column.
- Multiply the quotient times the Auxiliary number in the previous row and add the Auxiliary number in the row before that, putting the result in the Auxiliary column.
- When the remainder is reduced to 1, the content of the Auxiliary column in that row is the inverse of a(x).

By following the above process we get the inverse value of 0x0F which is 0x08. The multiplicative inverse table is given in Figure 12.

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	1	9	E	D	A	7	6	F	2	C	5	A	4	3	8

Figure 12: Multiplicative inverse table

5.3.2 Proposed S-Box generation methodology

Proposed AES S-box generation procedure is given below:

- $x^4 + x + 1$ is used as irreducible polynomial
- b[0]...b[3] is the multiplicative inverse of input
- Affine transformation
- The 4x4 matrix is formed using a formula which is given below:

$$b_i = b_i \; XOR \; b_{(i+2) \; mod \; 4} XOR \; b_{(i+3) \; mod \; 4} \; XOR \; C_i$$

$$b_i = i^{th} \; bit \; of \; input \; byte$$

$C_i = i^{th}$ *bit of a specially designated byte which is hexadecimal of* $1, 5, 11, 12, 15$ *as they don't generate any fixed points*

- Finally we get the output byte d[0]...d[7]

Figure 13 shows the overall procedure from top to bottom.

For the values of C ranging from 0 to 15 only for 5 values we will not get fixed point. Fixed point means for any input in the s-box we will get the same value as output. Here for hexadecimal value 1,5,11,12,15 we won t get any fixed point.

Different s-boxes and inverse s-boxes for different values of C is given below:

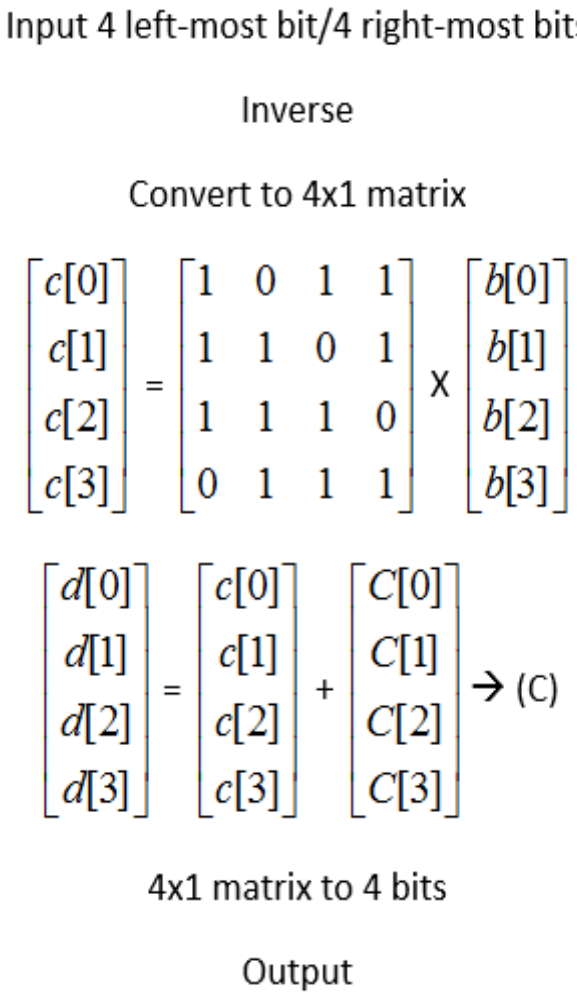

Figure 13: Proposed AES S-Box generation process

➢ **Case-1: When C = 0x01**

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	F	4	0	9	A	C	B	7	6	E	2	D	5	1	3

➢ **Inverse S-box:**

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
3	E	B	F	2	D	9	8	0	4	5	7	6	C	A	1

Figure 14: Case-1: When C = 0x01

➢ **Case-2: When C = 0x05**

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A	D	6	2	B	8	E	9	5	4	C	0	F	7	3	1

➢ **Inverse S-box:**

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
B	F	3	E	9	8	2	D	5	7	0	4	A	1	6	C

Figure 15: Case-2: When C = 0x05

➢ **Case-3: When C = 0x0B**

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
D	A	1	5	C	F	9	E	2	3	B	7	8	0	4	6

➢ **Inverse S-box:**

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
D	2	8	9	E	3	F	B	C	6	1	A	4	0	7	5

Figure 16: Case-3: When C = 0x0B

➢ **Case-4: When C = 0x0C**

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
3	4	F	B	2	1	7	0	C	D	5	9	6	E	A	8

➢ **Inverse S-box:**

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
7	5	4	0	1	A	C	6	F	B	E	3	8	9	D	2

Figure 17: Case-4: When C = 0x0C

➢ **Case-5: When C = 0x0F**

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
F	8	3	7	E	D	B	C	0	1	9	5	A	2	6	4

➢ **Inverse S-box:**

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	9	D	2	F	B	E	3	1	A	C	6	7	5	4	0

Figure 18: Case-5: When C = 0x0F

6 Implementation & Analysis

6.1 Current progress

As we are proposing a lightweight version of AES for RCE, first we analyze the actual mechanism of AES. For that purpose we have done the following tasks.

- Understanding the overall mechanism first we have implemented AES algorithm in C language.
- Implementing AES-128 Encryption in CC2420 radio chip of telosb.
- Sensing temperature from environment and sent the sensed data from one mote to other with encryption and without encryption and measured the voltage deviation.

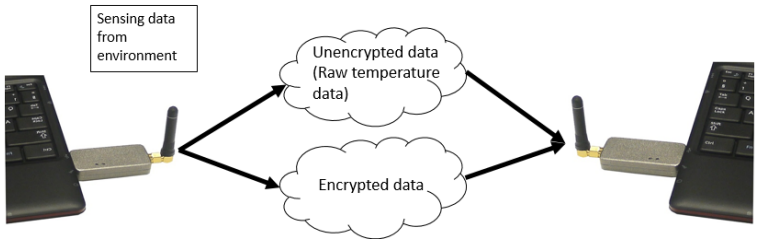

Figure 19: Data Sensing and transferring

From our C implementation we can see the AES encryption & decryption in Figure 20 and 21:

```
Enter your Cipher Key:
24 75 A2 B3 34 75 56 88 31 E2 12 00 13 AA 54 87

Enter your Plain Text:
00 04 12 14 12 04 12 00 0C 00 13 11 08 23 19 19

CipherText:
BC 02 8B D3 E0 E3 B1 95 55 0D 6D F8 E6 F1 82 41
```

Figure 20: AES Encryption

```
Enter your Cipher Key:
24 75 A2 B3 34 75 56 88 31 E2 12 00 13 AA 54 87

Enter your Cipher Text:
BC 02 8B D3 E0 E3 B1 95 55 0D 6D F8 E6 F1 82 41

PlainText:
00 04 12 14 12 04 12 00 0C 00 13 11 08 23 19 19
```

Figure 21: AES Decryption

After the modification of AES S-Box, we get the encryption & the decryption when C=0x01 which are shown in Figure 22 and 23:

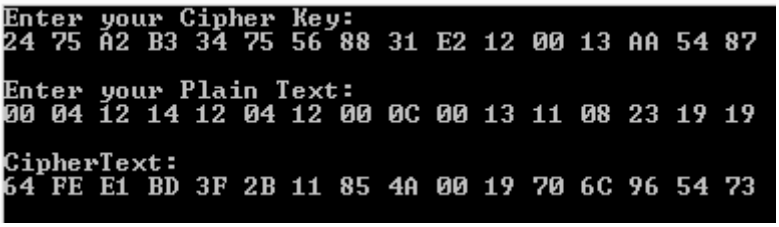

Figure 22: Modified AES Encryption

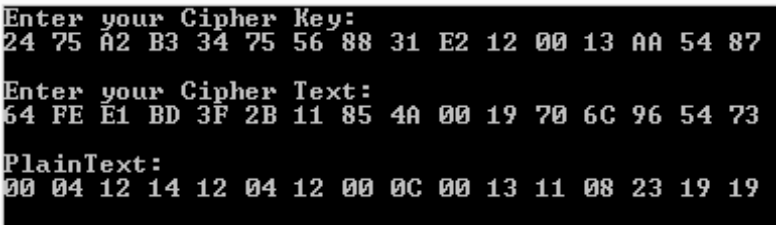

Figure 23: Modified AES Decryption

6.2 TinyOS implementation

We have already finished our implementation for both encrypted and unencrypted data for TinyOS. Figure 24 shows voltage comparison between values with and without encryption. X-axis represent the packet number and Y-axis represent the raw voltage data.

We also implemented our modified AES in TinyOS. Same as the previous one we took data from the environment through telosB sensor nodes and sent the data over the air from one node to another node. This time we encrypt the data using our modified AES and also using original AES. After taking data for 2 days we plot the data. From figure 25 we can see the comparison between modified AES and original AES. When the voltage is dropped from 3.50V to 3.01V, 18915 packets are sent from one sensor node to sink node when we use original AES algorithm for encryption. But in case of modified AES, 22385 packets are sent form same amount of voltage degradation. About 18.35% more packets can be sent over the air from the sender to the sink node. Our propsed algorithm gives much more energy efficiency then the original one. Figure 26 shows the efficiency of our proposed algorithm.

6.3 Strength of our proposal

Main strength of our modified AES algorithm is given below:

- We implemented our modified AES algorithm in C language
- It can encrypt the plain text into the cipher text using a cipher key and decrypt the cipher text into the actual plain text using the same cipher key.
- **Memory Complexity :** In terms of memory complexity our modified AES saves memory than original one as we used 1D S-box instead of 2D S-box.

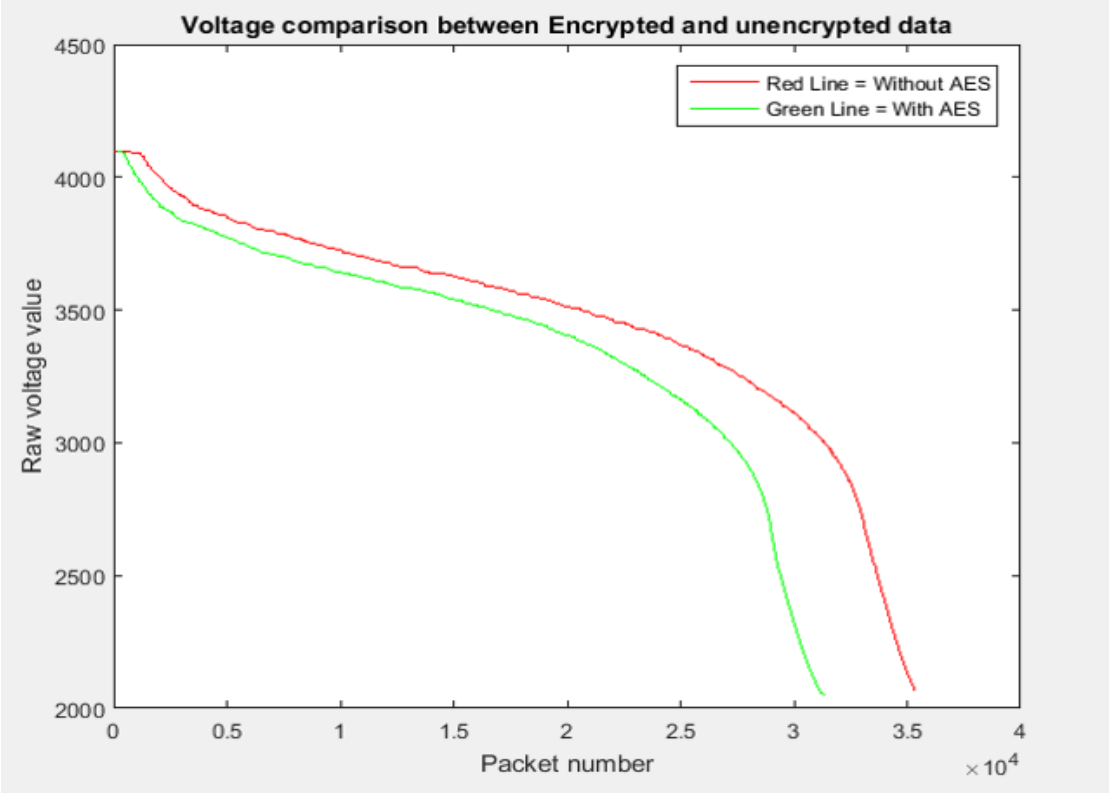

Figure 24: Voltage comparison between Encrypted and unencrypted data

7 Conclusion

In this paper, we proposed a modified version of AES which is suitable for IoT devices as it consumes less energy then the original one. We proposed a new Substitution Box for AES which works in GF(2^4) and much memory efficient in comparison to original AES Substitution Box. We analyze that while transmitting data(temperature) from one sensor mote to another without encrypting, consumes less energy then encrypted temperature data through plotting a graph. We also generate a graph which shows us that our proposed AES consumes less energy then the original AES which makes our proposal a lightweight version of AES and it can be used in Resource Constraint Environment.

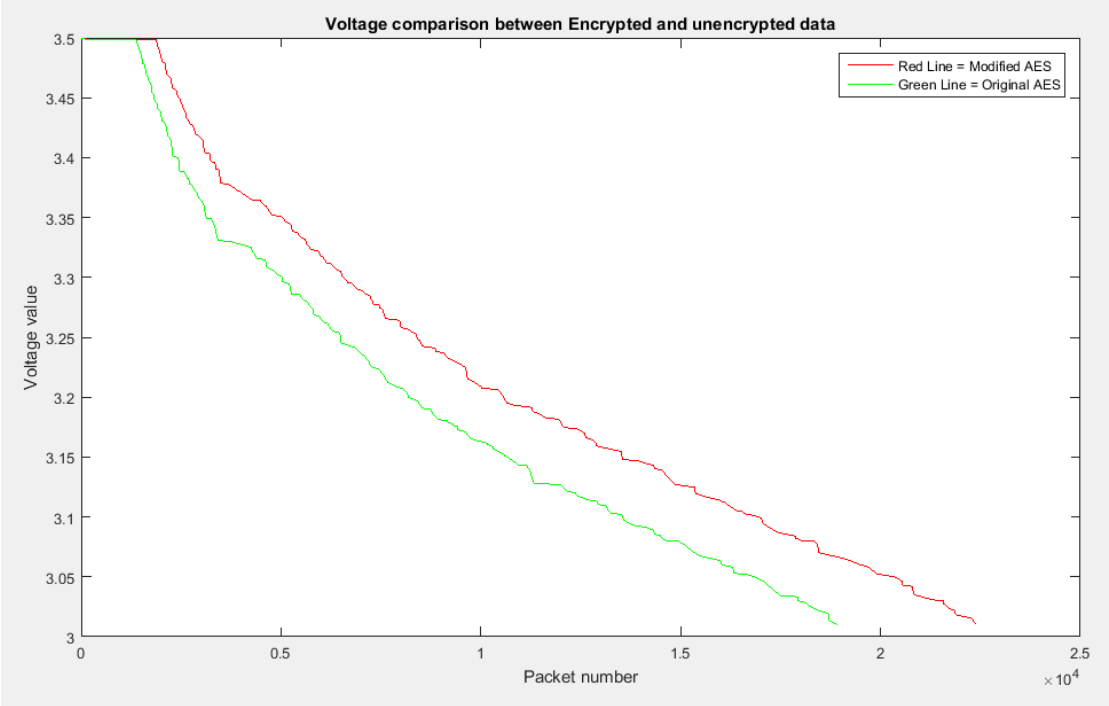

Figure 25: Voltage comparison between modified and original algorithm

Algorithm	Packets	Voltage degradation	Efficiency
Rijndael AES	18915	3.50 to 3.01	18.35%
Modified AES	22385	3.50 to 3.01	

Figure 26: Efficiency of modified AES algorithm

References

[1] A survey of lightweight-cryptography implementations, Eisenbarth, Thomas and Kumar, Sandeep and Paar, Christof and Poschmann, Axel and Uhsadel, Leif, IEEE Design & Test of Computers, 2007.

[2] PRINCE–a low-latency block cipher for pervasive computing applications Borghoff, Julia and Canteaut, Anne and Güneysu, Tim and Kavun, Elif Bilge and Knezevic, Miroslav and Knudsen, Lars R and Leander, Gregor and Nikov, Ventzislav and Paar, Christof and Rechberger, Christian and others, Springer, 2012.

[3] The SIMON and SPECK lightweight block ciphers, Beaulieu, Ray and Shors, Douglas and Smith, Jason and Treatman-Clark, Stefan and Weeks, Bryan and Wingers, Louis, ACM, 2015.

[4] The design of Rijndael: AES-the advanced encryption standard, Daemen, Joan and Rijmen, Vincent, Springer Science & Business Media, 2013.

[5] www.iwar.org.uk/comsec/resources/cipher/sha256-384-512.pdf

[6] AES implementation on a grain of sand, Feldhofer, Martin and Wolkerstorfer, Johannes and Rijmen, Vincent, IEE Proceedings-Information Security, 2005.

[7] Pushing the limits: a very compact and a threshold implementation of AES, Moradi, Amir and Poschmann, Axel and Ling, San and Paar, Christof and Wang, Huaxiong, Springer, 2011.

[8] Harvesting the potential of nano-CMOS for lightweight cryptography: an ultra-low-voltage 65 nm AES coprocessor for passive RFID tags, Hocquet, Cédric and Kamel, Dina and Regazzoni, Francesco and Legat, Jean-Didier and Flandre, Denis and Bol, David and Standaert, François-Xavier, Springer, 2011.

[9] Towards green cryptography: a comparison of lightweight ciphers from the energy viewpoint, Kerckhof, Stéphanie and Durvaux, François and Hocquet, Cédric and Bol, David and Standaert, François-Xavier, Springer, 2012.

[10] Dietary recommendations for lightweight block ciphers: power, energy and area analysis of recently developed architectures, Batina, Lejla and Das, Amitabh and Ege, Barış and Kavun, Elif Bilge and Mentens, Nele and Paar, Christof and Verbauwhede, Ingrid and Yalçın, Tolga, Springer, 2013.

[11] Exploring energy efficiency of lightweight block ciphers, Banik, Subhadeep and Bogdanov, Andrey and Regazzoni, Francesco, proceedings of SAC, 2015.

[12] Exploring the energy consumption of lightweight blockciphers in FPGA, Banik, Subhadeep and Bogdanov, Andrey and Regazzoni, Francesco, IEEE, 2015.

[13] A comprehensive survey of modern symmetric cryptographic solutions for resource constrained environments, Kong, Jia Hao and Ang, Li-Minn and Seng, Kah Phooi, Elsevier, 2015.

[14] Compact FPGA implementation of PRESENT with Boolean S-Box, Tay, JJ and Wong, MLD and Wong, MM and Zhang, C and Hijazin, I, IEEE, 2015.

[15] Energy efficient security architecture for wireless sensor networks, Zivkovic, Milan and Branovic, I and Markovic, Dejan and Popovic, R, IEEE, 2012.

[16] Modified AES Algorithm Using Multiple S-Boxes, Felicisimo V. Wenceslao, Jr, Bobby D. Gerardo, Bartolome T. Tanguilig III, Second International Conference on Electrical, Electronics, Computer Engineering and their Applications (EECEA2015), 2015.