Internet of Things (IoT) Security

Shivaji Kulkarni
Dept. of Electronics and Communication, B.V.B. College of Engineering and Technology, Hubli, INDIA
Email ID: shivajikulkarni123@gmail.com

Shrihari Durg
Dept. of Electronics and Communication, B.V.B. College of Engineering and Technology, Hubli, INDIA
Email ID: shrihari.durg@gmail.com

Nalini Iyer
Dept. of Electronics and Communication, B.V.B. College of Engineering and Technology, Hubli, INDIA
Email ID: nalinic@bvb.edu

Abstract – Network Security is one of the important concepts in data security as the data to be uploaded should be made secure. To make data secure, there exist number of algorithms like AES (Advanced Encryption Standard), IDEA (International Data Encryption Algorithm) etc. These techniques of making the data secure come under Cryptography. Involving Internet of Things (IoT) in Cryptography is an emerging domain. IoT can be defined as controlling things located at any part of the world via Internet. So, IoT involves data security i.e. Cryptography. Here, in this paper we discuss how data can be made secure for IoT using Cryptography.

Keywords – Network Security; AES; Cryptography; IoT.

NOMENCLATURE

IoT – Internet of Things.

AES - Advanced Encryption Standard

I. INTRODUCTION

IoT Security involves securing the data and uploading the data to cloud. Using the AES-128 algorithm we encrypt the data. AES involves 4 steps which are –

- Substitute byte or S-box.
- Shift rows.
- Mix Column.
- Add round key.

These steps are carried out for 10 rounds. After the 10^{th} round we get the encrypted data. The data is uploaded to the cloud using an IoT (Internet of Things) development board. Here we are using Intel Galileo board. We can retrieve the data at the cloud by using inverse AES-128 algorithm which involves the steps as shown –

- Inverse Substitute bytes or Inverse S-box.
- Inverse Shift rows.
- Inverse Mix Column.
- Inverse Add round key.

Employing these steps for 10 rounds gives the decrypted data i.e. the original data. Fig. 1 below gives the block diagram overview.

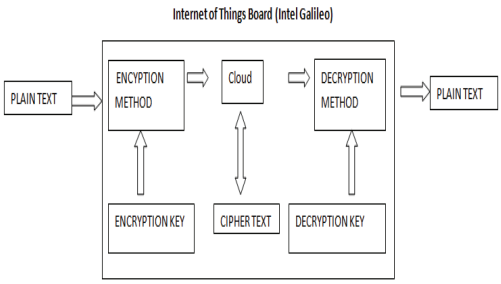

Fig. 1. Block Diagram of IoT Security

II. AES-128 ALGORITHM

Advanced Encryption Algorithm is also known as Rijndael was established by U.S. National Institute of Standards and Technology (NIST) in 2001. AES uses 3 types of data which are –

- 128 bits with 10 rounds.
- 192 bits with 12 rounds.
- 256 bits with 14 rounds.

AES-128 uses 128 bit plain text. There are 10 rounds with each round having 4 steps. Since it is a 128 bit data, we represent the data in a 4x4 matrix with each element being 8 bits. Each round has its own key which is also represented by a 4x4 matrix with each element being 8 bits. For data encryption, input is plaintext data and the output is cipher data or encrypted data. For data decryption, input would be cipher text data and the output is original or plain text data. The following explains about the 4 steps involved in the algorithm.

A Substitute bytes or S-box

Each element in the plain text data is replaced by another element present in the S-box. S-box is a 16x16 matrix with each element of 8 bits. Each element is a hexadecimal number. Each element in the plain text is replaced by an element in the S-box in such a way that the first 8 bits refer to the row and the next 8 bits refer to the column. The S-box usually used in shown in Fig. 2.

978-9-3805-4421-2/16/$31.00 ©2016 IEEE

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

Fig. 2. S-Box

Example – If the plain text data is 0xc1, it would be replaced by the element in the S-box located at 'c' th row and '1' st column. Thus, the element 0xc1 is replaced by 0x78. A clear representation is shown in Fig. 3.

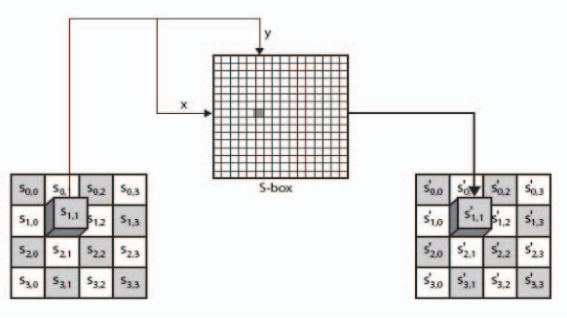

Fig. 3. S-Box representation

Thus, the output of this step would be a 4x4 matrix of 8 bits each.

B Shift Rows

The input to this step is the output of Substitute byte step. This step applies to the rows of the matrix. The first row remains unchanged. The second row gets left shifted by 1 cyclically. The third row gets left shifted by 2 cyclically and the fourth row gets left shifted by 3 cyclically. Fig. 4 represents the shift row step.

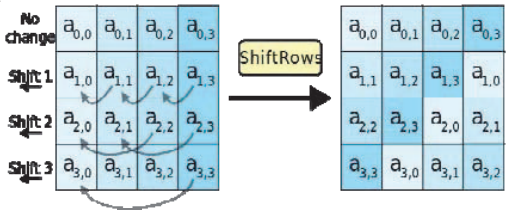

Fig. 4. Shift Row Overview

The output of this step is a 4x4 row shifted matrix with each element of 8 bits.

C Mix Column

The input to this step is a 4x4 row shifted matrix. This is an important step. It involves matrix multiplication with a polynomial GF $(2^8) = x^8 + x^4 + x^3 + x + 1$. Using this polynomial we can have a matrix c(x) which is a 4x4 matrix of 8 bits elements. The matrix c(x) usually used is shown in Fig. 5.

$$c(x) = \begin{bmatrix} 0x02 & 0x03 & 0x01 & 0x01 \\ 0x01 & 0x02 & 0x03 & 0x01 \\ 0x01 & 0x01 & 0x02 & 0x03 \\ 0x03 & 0x01 & 0x01 & 0x02 \end{bmatrix}$$

Fig. 5. c(x) matrix

Multiplication by 0x01 makes no difference in the value. Thus, the value would remain the same.

Multiplication by 0x02 means shifting the data towards left by 1 bit. If the original data (data before shift) had a high bit at MSB (Most Significant Bit) then the shifted data needs to be XORed with 0x1b.
Example – data = 0x11.
After multiplication by 0x02, the value would be 0x22. 0x11=0001 0001.
After left shift by 1 bit value is 0010 0010. Make LSB (Least Significant Bit) as 0.
Since there is no high bit for 0x11 at MSB there is no XOR with 0x1b.

Multiplication by 0x03 means shifting the data towards left by 1 bit and having a XOR with the original data i.e. the data before shifting. If the original data (data before shift) had a high bit at MSB (Most Significant Bit) then the shifted data needs to be XORed with 0x1b.
Example - data = 0x87.
After multiplication by 0x03, the value would be 0x92. 0x87=1000 0111.
After left shift by 1 bit and XOR with 0x87 we have 0x89. (1000 0111 XOR 0000 1110 = 1000 1001)
Since the MSB in original data 0x87 is high, we need to XOR it with 0x1b. Thus, the result is 0x92 (1000 1001 XOR 0001 1011).
Fig. 6 gives a better view of Mix Column step.

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

Fig. 6. Mix Column Step

where
$S_{0,0}^{1}$ = (0x02 * $S_{0,0}$) XOR (0x0x3 * $S_{1,0}$) XOR (0x01 * $S_{2,0}$) XOR (0x01 * $S_{3,0}$)
$S_{0,1}^{1}$ = (0x02 * $S_{0,1}$) XOR (0x0x3 * $S_{1,1}$) XOR (0x01 * $S_{2,1}$) XOR (0x01 * $S_{3,1}$)

D Add Round Key

This is the final step involved in the algorithm. The input to this step is the Mix Column matrix. There are keys derived for each round from the original key. Having a XOR operation of Mix Column matrix with keys generated for corresponding round gives the output for the current round. The output of this round would be an input to the next round. The output of this step after 10 rounds provides the encrypted data. The key for each round is derived from the original key using Rijndael's key schedule.

Note – Mix Column step is skipped for the 10^{th} round.

III. INVERSE AES-128 ALGORITHM

The input to the inverse algorithm is the encrypted data and the output is plain text or original data. The inverse AES-128 algorithm involves the following steps –

A Inverse Substitute bytes or Inverse S-box

It works the same as that of Substitute bytes but instead has an inverse S-Box. This step provides an output 4x4 matrix with each element of 8 bits. The inverse S-box usually used is shown in Fig. 7.

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

Fig. 7. Inverse S-box

B Inverse Shift Rows

The input to this step is the output of Inverse Substitute bytes step. This step applies to rows. The first row is kept the same. The second row is shifted towards 1 by right cyclically. The third row is shifted rightwards by 2 cyclically. The fourth row is cyclically right shifted by 3.

C Inverse Mix Column

The step is similar to Mix Column step. The matrix c(x) changes and is shown in Fig. 8.

Multiplication by 0x09 -

This can be done by splitting 09 as 2*4 + 1. So, it is multiplication by 2 four times and XOR with the original data.

Multiplication by 0x0b -

This can be done by splitting 0b as 2*5 + 1. It would be multiplication by 2 five times. The result is XORed with original data.

Multiplication by 0x0d –

We split 0d as 2*6 + 1. A XOR of original data multiplied by 2 six times with the original gives the result.

Multiplication by 0x0e –

We split 0e as 2*7. The result would be a XOR of original data multiplied by 2 seven times.

D Inverse Add Round Key

The output of inverse Mix Column is an input to this step. Inverse keys are generated for each round using the original key. A XOR of keys for the corresponding round with the inverse mix column matrix gives an output. This output is an input to the next round. Again key generation here involves Rijndael's key schedule.

Note – Inverse Mix Column is skipped for 10^{th} round.

$$c(x) = \begin{bmatrix} 0x0e & 0x0b & 0x0d & 0x09 \\ 0x09 & 0x0e & 0x0b & 0x0d \\ 0x0d & 0x09 & 0x0e & 0x0b \\ 0x0b & 0x0d & 0x09 & 0x0e \end{bmatrix}$$

Fig. 8. Inverse Mix Column Step

IV. INTERNET OF THINGS (IOT) BOARD

An IoT board is a one through which we can upload data to the cloud. It has features like Ethernet, Wi-Fi etc. There are a number of IoT boards. Some of them are Intel Galileo, Raspberry Pi etc. We are using Intel Galileo Gen-2.

Features of Intel Galileo –

- A 32 bit microcontroller board based on Intel Quark SOC X1000 Application processor.
- It has memory of 256 MB.
- It has 14 input/output digital pins.
- It has 6 analog pins via an A-D-C (Analog to Digital Convertor).
- It works on Linux Operating System and is compatible Arduino IDE (Integrated Development Environment).
- It supports Ethernet, USB device connectors and optional SD card.

V. WORKING

The input data from any source is given to the AES-128 algorithm. It provides a secured cipher text. This encrypted data is uploaded to the cloud via the Intel Galileo Gen 2. The data can be retrieved from the cloud and original data is obtained by using the inverse AES-128 algorithm.

VI. APPLICATION

Bird Hit Probability -

In Bird Hit Probability, using a camera we capture the images of the birds. Using image processing we locate the birds position in space. The location includes x, y and z co-ordinates. This x, y and z co-ordinates is the main data. This acts as plain text. Using the AES-128 algorithm we encrypt the data and upload it to the cloud using Intel Galileo. This is shown in Fig. 9.

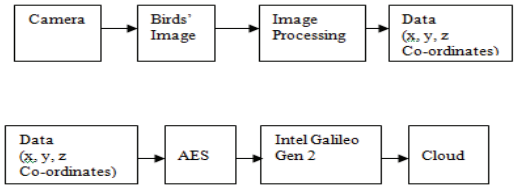

Fig. 9. Uploading encrypted data to Cloud

At the other end, data is retrieved and original data is obtained back using the AES-128 decryption algorithm, as shown in Fig. 10.

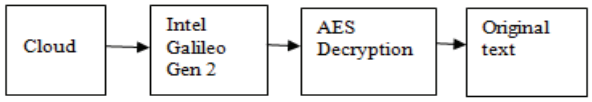

Fig. 10. Data retrieval from Cloud using inverse AES

During the travel of airplanes, birds flying in the air may collide with the airplane leading to damages. It may not only hurt the bird but also makes the flight unstable risking the lives of the travelers. We can avoid this by employing Bird Hit Probability. The decrypted data can be used by the pilot to deviate the plane slightly depending on the position of the bird.

VII. RESULTS AND OBSERVATIONS

Fig. 11 shows the image of a bird captured when detected.

Fig. 11. Image when bird is detected

Fig. 12 shows the data in terms of x, y and z co-ordinates.

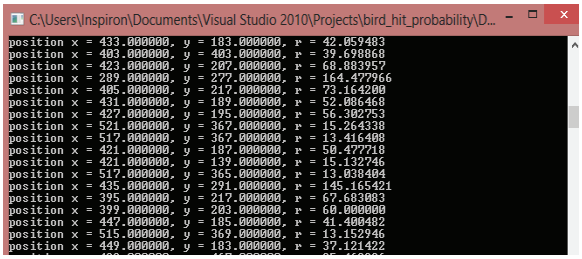

Fig. 12. Data in terms of x, y and z co-ordinates

VIII. CONCLUSION AND FUTURE SCOPE

The data encryption and decryption is done for 128 bits. To make the data more secure, we can use AES-192 or AES- 256 bits. In Bird Hit Probability, since the movement of birds is not stationary, capturing the image, processing it, encrypting it and uploading it to the cloud would consume time. This is a limitation. As a part of future scope, we can make the IoT board, a black box which would receive the plain text as input and gives encrypted output which can be used for many IoT Applications. By making this, even common men can secure the data.

APPENDIX

The Cryptography and Internet of Things is an emerging domain. Research and Development is under progress in this field. In this paper we have explored a part of this trending field by using an IoT Application named Bird Hit Probability.

ACKNOWLEDGEMENT

We would like to thank B. V. B. College of Engineering and Technology, Hubli for providing us an opportunity to research on Cryptography and IoT Security.

REFERENCES

[1]. Hui Suo, Jiafu Wan, Caifeng Zou, Jianqi Liu, "Security in the Internet of Things: A Review", *Computer Science and Electronics Engineering (ICCSEE)*, Volume: 3, ISBN: 978-1-4673-0689-8.

[2]. Ukil, A, Sen, J and Koilakonda, S, "Embedded security for Internet of Things", *Emerging Trends and Applications in Computer Science (NCETACS)*, ISBN: 978-1-4244-9578-8.

[3]. Simone Cirani, Gianluigi Ferrari and Luca Veltri, "Enforcing Security Mechanisms in the IP-Based Internet of

[4]. Things: An Algorithmic Overview", *Algorithms 2013*, 6, 197-226; doi:10.3390/a6020197.

[5]. Sumedha Kaushik and Ankur Singhal, "Network Security Using Cryptographic Techniques", *International Journal of Advanced Research in Computer Science and Software Engineering,* Volume 2, Issue 12, December 2012.

[6]. AK Srivastava, "Internet of Things and its enhanced data security", *International Journal of Engineering and Applied Sciences (IJEAS)*, ISSN: 2394-3661, Volume-2, Issue-2, February2015.

[7]. http://www.nayuki.io/page/aes-cipher-internals-in-excel

[8]. https://en.wikipedia.org/wiki/Advanced_Encryption_Standard

[9]. http://www.cse.wustl.edu/~jain/cse571-11/

[10]. http://www.journals.elsevier.com/ad-hoc-networks/call-for-papers/special-issue-on-internet-of-things-security-and-privacy/

[11]. Cryptography and Network Security by William Stallings.