

Implementation and Performance Analysis of AES-128 CBC algorithm in WSNs

Hyeopgeon Lee*, Kyoungghwa Lee**, Yongtae Shin*

**Department of Computingt, Soongsil University, Korea*

hlee@cherry.ssu.ac.kr, khlee@cherry.ssu.ac.kr, shin@ssu.ac.kr

Abstract— The sensor networks is a network technique for the implementation of Ubiquitous computing environment. It is wireless network environment that consists of the many devices, called the sensor node, which have miniature computing device, small memory space and very limited battery power. Though sensor network provides various capabilities, it is unable to ensure the secure authentication between sensor nodes. Eventually it causes the losing reliability of the entire network and many secure problems. Therefore, Symmetric key encryption algorithm with low-Energy consumption is required to the applicable sensor networks. In this paper, we proposed the solution of reliable sensor networks to analyze the communication efficiency through measuring performance of AES-128 CBC algorithm which is selected by default in sensor networks by plaintext size and cost of operation per hop according to the network scale.

Keywords— Wireless Sensor Networks, AES algorithm, CBC, Microcontroller Performance, AVR, JTAG

I. INTRODUCTION

The sensor networks is a network technique for the implementation of Ubiquitous computing environment. It is wireless network environment that consists of the many sensor nodes of lightweight and low-power. It is researching and developing at the various standards and research organizations. As a result, the sensor networks can be used such applications as environmental control, smart home network, logistics, military and so on [1]. In these environments, the data is collected by sensor nodes are used through the systematic analysis and the cross-linking between services in a variety of services. Therefore, common security requirements (integrity, confidentiality, authentication, non-repudiation) are required for security service and applications.

Public key encryption algorithm is a fundamental and widely using technology around the world. But it has hardware limitations as like memory and battery, so it is not applied to the sensor networks [2]. Therefore, Symmetric key encryption algorithm with low-Energy consumption is used in the sensor networks.

In this paper, we describe the AES-128 CBC algorithm in the symmetric key encryption which is selected by default in

sensor networks. And we measure the encryption and decryption performance on the 8-bit Microcontroller. Then, we analyse the communication efficiency through the total delay per hop in sensor networks. The structure of the paper is organized as follows: Section 2 describes The AES algorithm in Symmetric key encryption, round transformation and CBC mode; Section 3 measures the encryption and decryption performance on the 8-bit Microcontroller; Sections 4 analyzes the communication efficiency in sensor networks through the total delay per hop; and Section 5 concludes this paper.

II. SYMMTRIC KEY ENCRYPTION

A. AES(Advanced Encryption Standard)

The AES (advanced encryption standard) [3] [4], developed to replace the old Data Encryption Standard (DES), is an encryption standard as a symmetric block cipher. It was announced by National Institute of Standards and Technology (NIST) as U.S. Federal Information Processing Standard PUB 197 (FIPS 197) on November 26, 2001. The central design principle of the AES algorithm is the adoption of symmetry at different platforms and the efficiency of processing. After a 5-year standardization process, the NIST adopted the Rijndael algorithm as the AES.

The AES operates on 128-bit blocks of data which are length of the input block, the output block and the State. The algorithm can encrypt and decrypt blocks using secret keys. The key size can either be 128 bit, 192 bit, or 256 bit. The actual key size depends on the desired security level. The different versions are most often denoted as AES-128, AES-192 or AES-256. Table 1 shows that Key Block Round combinations in this standard.

Table 1. Key Block Round Combinations

	Key Length (N_k words)	Block Size (N_b words)	Number of Rounds (N_r)
AES-128	4	4	10
AES-192	5	4	12
AES-256	8	4	14

The AES consists of an initial Round Key addition, first Nr-1 Rounds, a final round. Figure 1 shows the pseudo code of AES algorithm.

```

Cipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
Begin
    byte state[4, Nb]
    state = in
    AddRoundKey(state, w[0, Nb-1])
    for round = 1 step 1 to Nr-1
        SubBytes(state)
        ShiftRows(state)
        MixColumns(state)
        AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
    end for
    SubBytes(state)
    ShiftRows(state)
    AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])
    out = state
end

```

Figure 1. Pseudo code of AES algorithm

The key expansion can be done on beforehand and AES can be specified in terms of the Expanded Key. The Expanded Key shall always be derived from the Cipher Key and never be specified directly. There are however no restrictions on the selection of the Cipher Key itself.

Figure 2 shows the pseudo code of AES's Expanded Key algorithm.

```

KeyExpansion(byte key[4*Nk], word w[Nb*(Nr+1)], Nk)
begin
    word temp
    i = 0
    while (i < Nk)
        w[i] = word(key[4*i], key[4*i+1], key[4*i+2], key[4*i+3])
        i = i+1
    end while
    i = Nk
    while (i < Nb * (Nr+1))
        temp = w[i-1]
        if (i mod Nk = 0)
            temp = SubWord(RotWord(temp)) xor Rcon[i/Nk]
        else if (Nk > 6 and i mod Nk = 4)
            temp = SubWord(temp)
        end if
        w[i] = w[i-Nk] xor temp
        i = i + 1
    end while
end

```

Figure 2. Pseudo code of AES's Expanded Key algorithm

B. Round transformation of AES

The round transformation [5] modifies the 128-bit State. The initial State is the input plaintext and the final State is the output ciphertext. The State is organised as a 4 X 4 matrix of bytes. The round transformation scrambles the

bytes of the State either individually, rowwise, or columnwise by applying the functions SubBytes, ShiftRows, MixColumns, and AddRoundKey sequentially. Figure 3 show the AES-128 iterates a round transformation. Figure 3 show the AES iterates a round transformation.

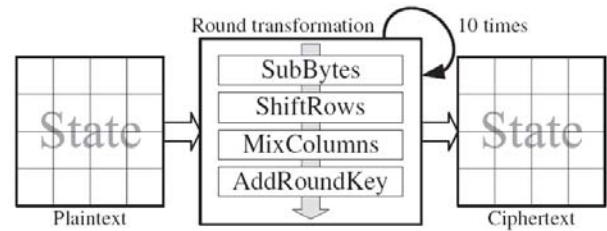


Figure 3. AES-128 iterates a round transformation.

An initial AddRoundKey operation precedes the first round. The last round differs slightly from the others the MixColumns operation is omitted.

1) SubByte

SubByte [3] is a substitution function in the Cipher round. In the SubBytes step, each byte in the state is replaced with its entry using a nonlinear byte substitution table (S-box) that operates on each of the State bytes independently. Figure 4 shows the SubBytes applies the S-box to each byte of the State.

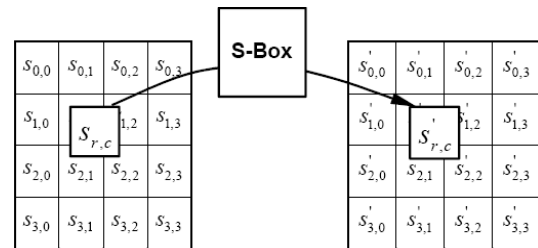


Figure 4. SubBytes applies the S-box to each byte of the State

2) ShiftRows

ShiftRows [3] is a permutation function in the Cipher round. In the ShiftRows step, bytes in each row of the state are shifted cyclically to the left. The number of places each byte is shifted differs for each row. ShiftRows step is composed of bytes from each column of the input state. Figure 5 show the ShiftRows cyclically shifts the last three rows in the State.

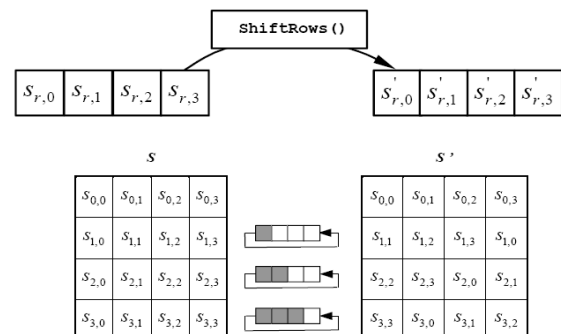


Figure 5. ShiftRows cyclically shifts the last three rows in the State

3) MixColumns

MixColumns [3] is a Mixing function in the Cipher round. In the MixColumns step, the four bytes of each column of the state are combined using an invertible linear transformation. The MixColumns function takes four bytes as input and outputs four bytes, where each input byte affects all four output bytes. Together with ShiftRows, MixColumns provides diffusion in the Cipher. Figure 6 shows the MixColumns operates on the State column-by-column.

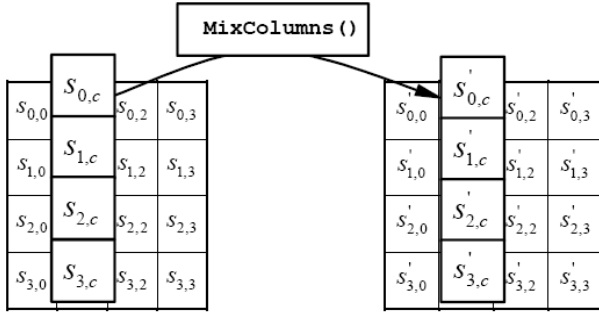


Figure 6. MixColumns operates on the State column-by-column

4) AddRoundKey

AddRoundKey [3] is a key adding function in the Cipher round. In the AddRoundKey step, the subkey is combined with the state. For each round, a subkey is derived from the main key using Rijndael's key schedule, each subkey is the same size as the state. The subkey is added by combining each byte of the state with the corresponding byte of the subkey using bitwise XOR. Figure 7 shows the AddRoundKey XORs each column of the State with a word from the key schedule.

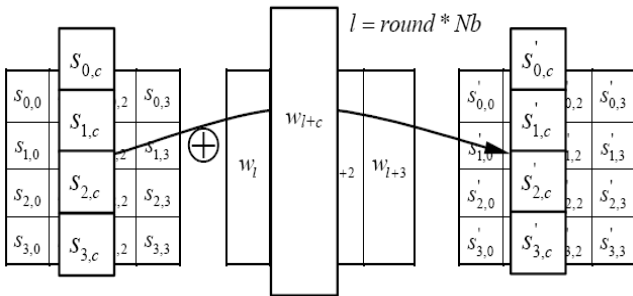


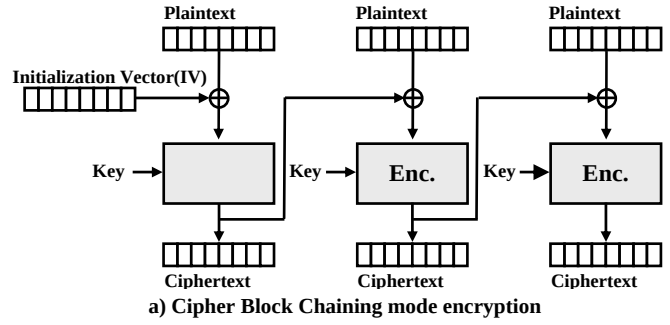
Figure 7. AddRoundKey XORs each column of the State with a word from the key schedule

AES Decryption computes the original plaintext of an encrypted ciphertext. During the decryption, the AES algorithm reverses encryption by executing inverse round transformations in reverse order. The round transformation of decryption uses the functions AddRoundKey, InvMixColumns, InvShiftRows, and InvSubBytes.

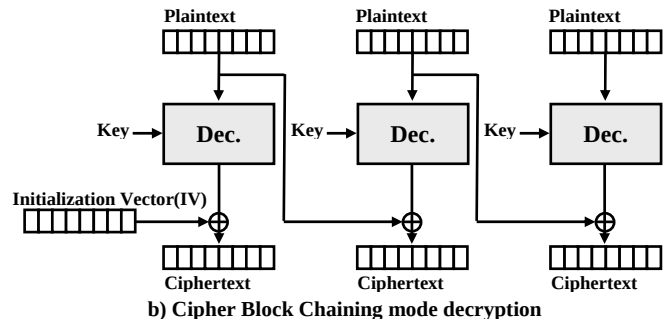
C. CBC (Cipher Block Chaining) mode

The CBC (Cipher Block Chaining) [6] [7] uses feedback to feed the result of encryption back into the encryption of

the next block. In CBC mode, each block of plaintext is XORed with the previous ciphertext block before being encrypted. This way, each ciphertext block is dependent on all plaintext blocks processed up to that point. Also, to make each message unique, an IV (initialization vector) must be used in the first block. The IV does not have to be kept secret. The IV should be a random number (or a serial number), to ensure that each message is encrypted uniquely. Figure 8 shows that CBC mode encryption and decryption.



a) Cipher Block Chaining mode encryption



b) Cipher Block Chaining mode decryption

Figure 8. Cipher Block Chaining mode encryption and decryption

If the first block has index 1, CBC encryption is the following:

$$C_i = E_K(P_i \oplus C_{i-1}), C_0 = IV \quad (1)$$

CBC decryption is the following:

$$P_i = D_K(C_i \oplus C_{i-1}), C_0 = IV \quad (2)$$

CBC has been the most commonly used mode of operation. Its main drawbacks are that encryption is sequential (i.e., it cannot be parallelized), and that the message must be padded to a multiple of the cipher block size. One way to handle this last issue is through the method known as ciphertext stealing.

Note that a one-bit change in a plaintext affects all following ciphertext blocks. A plaintext can be recovered from just two adjacent blocks of ciphertext. As a consequence, decryption can be parallelized, and a one-bit change to the ciphertext causes complete corruption of the corresponding block of plaintext, and inverts the corresponding bit in the following block of plaintext.

III. IMPLEMENTATION AND PERFORMANCE ANALYSIS

A. Device and Experiment environment

For the performance analysis of AES encryption algorithm in the sensor networks, we use the ATmega644p in 8-bit Microcontroller as a hardware device. The AVR Studio 4 and Programmer's Notepad in the WinAVR are used as development tools. The JTAG (Joint Test Action Group) Emulator is used as a debugging tool. Figure 9 shows that device for the performance analysis of the AES encryption algorithm.

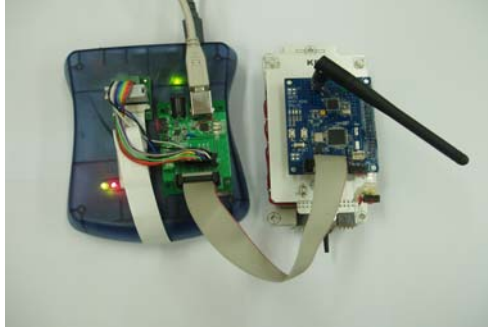


Figure 9. Device for the performance analysis of AES

1) ATmega644p Microcontroller

ATmega644p [8] in 8-bit Microcontroller is made by Atmel. It is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. And it has 64K bytes of In-System Programmable Flash with Read-While-Write capabilities, 2K bytes EEPROM, 4K bytes SRAM, 20Mhz System Clock and prescaler of 8, 64, 256 or 1024. The main function of the ATmega644p is to ensure correct program execution. It must therefore be able to access memories, perform calculations, control peripherals, and handle interrupts.

The ATmega644P is supported with a full suite of program and system development tools including: C compilers, macro assemblers, program debugger/simulators, in-circuitemulators, and evaluation kits. Figure 10 shows that Block Diagram of the ATmega644p Architecture.

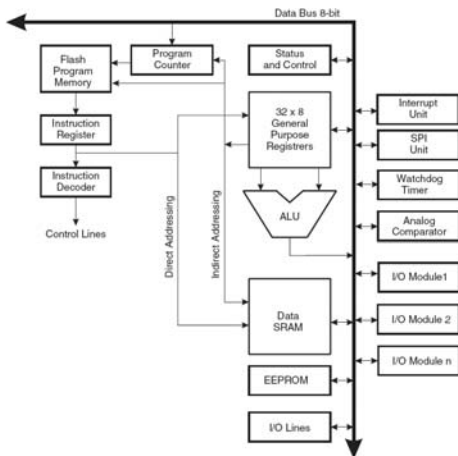


Figure 10. Block Diagram of the ATmega644p Architecture

2) AVR Studio

AVR Studio is an Integrated Development Environment for writing and debugging AVR applications. It provides a project management tool, source file editor and chip simulator. It also interfaces with In-Circuit Emulators and development boards available for the AVR 8-bit RISC family of microcontrollers. And compiled programs are applied to the AVR. Programmer's Notepad with the Win-GCC Compiler compiles the written C language. The compiled programs are applied to the AVR Studio.

3) JTAG Emulator

JTAG Emulator in JTAG Standard is I/O device using JTAG Port which receives the information from PCB or IC. And it uses the standard JTAG interface to enable the user to do real-time emulation of the microcontroller while it is running in the target system.

B. The implementation of principle

For the performance Measurement of AES encryption algorithm, we apply the AES-128 CBC mode to the ATmega644p's EEPROM.

The timer mode for the time measurement uses the Timer/Counter CTC (Clear Timer on Compare Match) Mode. The CTC Mode generates the compare interrupt only if the counter value (TCNT), which is cleared to zero, matches the OCR. The timer measurement measures the counts (P) of the compared interrupt per 1ms. Figure 11 shows that the timer measurement using Timer/Counter CTC Mode.

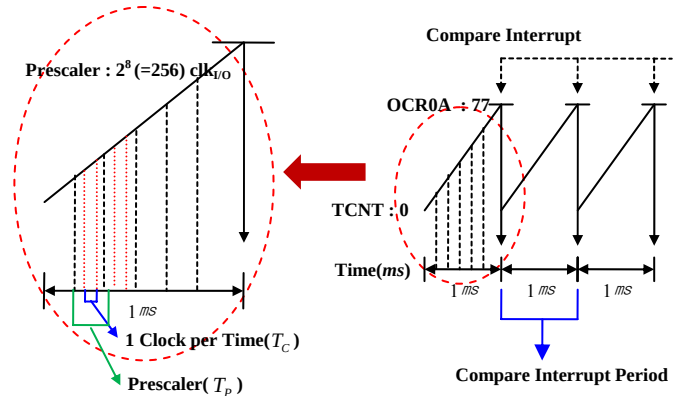


Figure 11. The timer measurement using Timer/Counter CTC Mode

The operation time per 1 clock (T_c) is the following:

$$T_c = \frac{1}{\text{Frequency}} = \frac{1}{SC * 10^6} \quad (3)$$

The ATmega644P has a system clock prescaler, and the system clock (SC) can be divided by setting the Clock Prescale Register. The prescale time per system clock prescaler (T_p) is the following:

$$T_p = \text{prescaler} * T_c \quad (4)$$

The Timer/Counter (TCNT) and Output Compare Registers (OCR) are 8-bit Registers. The OCR for the generating of the compare Interrupt is the following:

$$OCR0A = 0xFF - (0xFF - (P / T_p) + 1) \quad (5)$$

C. Result

For the comparison between encryption and decryption performance, we use the AES-128 CBC mode. The operation time of the encryption and decryption is measured to the data sizes of 16, 32, 64, 128, 256 and 512 Byte. Table 2 and Figure 12 show that the encryption and decryption operation time and CPU cycle according to the data size.

Table 2. the comparison between encryption and decryption performance by data sizes

Data Size(byte)		16	32	64	128	256	512
Enc.	Time (ms)	449	898	1,796	3,592	7,184	14,368
	CPU Cycle	8,980	17,960	35,920	71,840	143,680	287,360
Dec.	Time (ms)	456	912	1,825	3,649	7,297	14,592
	CPU Cycle	9,120	18,240	36,500	72,980	145,940	291,840

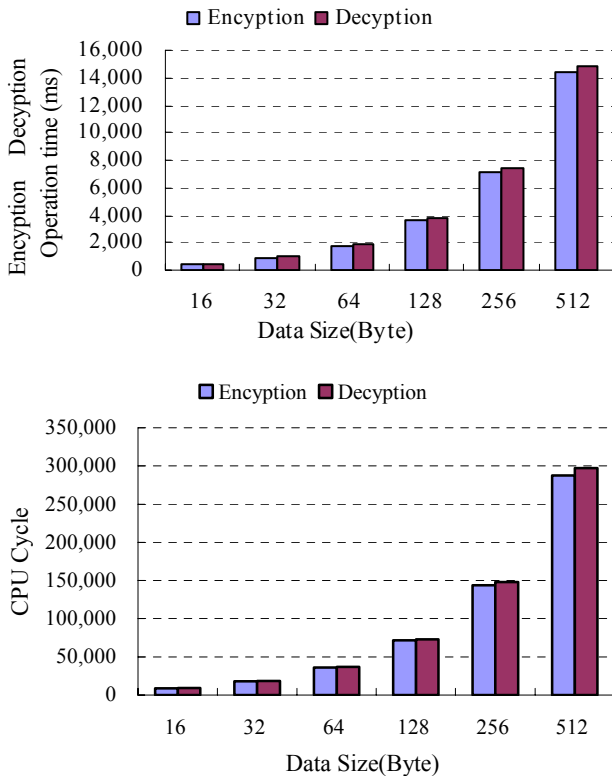


Figure 12. The operation time and CPU Cycle by data sizes

In the result, the operation time and CPU Cycle by data sizes increase approximately 2 times. In 512 byte, it takes approximately 14 minutes to the encryption and decryption.

IV. APPLICATION SCENARIO

A. Network model

Figure 13 shows that a general node (N_i) sends the secured data packet to the cluster head (CH) in the same subnet.

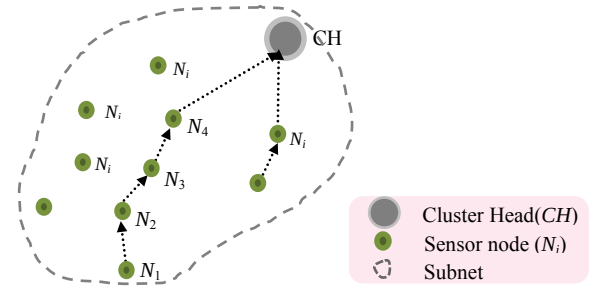


Figure 13. Sensor Networks Application Model

For measurement of the data encryption and decryption transmission delay by the number of communication hop, the following assumptions are established. Namely, the every node within subnet has same performance, and there is no interfere or packet loss in the data communication. Each node shares common key with neighbor nodes in advance, and operates encryption and decryption once per hop. The communication for the generating of Pair-wise Shared Key is similar to the μ TESLA (Micro Timed Efficient Stream Loss-tolerant Authentication) [9] protocol of the sensor networks.

B. Communication delay in sensor networks

In communication process of the sensor networks, the Beacon Request Command and Association Request Command are communicated between new node and cluster head. The general node (N_1) encrypts the data using the pre-deployed security key. It sends secured data to the neighbor node (N_2). The node (N_2) decrypts the encrypted message (msg_E) using the pre-deployed security keys. Then it obtained to the plaintext. The node (N_2) repeats the same process in the previous step using the private key shared with its neighbor node (N_3).

The data delivery process by hop communication is the following:

$$\begin{aligned} \forall N_i \in subnet \quad (i = 0 \dots n) \\ N_i \rightarrow N_{i+1} : msg_E = E < K_{prv}, plaintext > \\ N_{i+1} \rightarrow N_{i+n} : msg_E = E < K_{prv}, D < K_{prv}, msg_E >> \end{aligned} \quad (6)$$

If the delay by hop communication includes encryption delay, decryption delay and data transfer delay, total delay is the following:

$$T_{hop-by-hop} = t_{Enc} + t_{Transmission} + t_{Dec} + \Delta t \quad (7)$$

The Δt in equation (7) represents the delay for the allocation and channel access. It has between zero and $T_{hop-by-hop}$. When the general node and the cluster head communicate to the encrypted packet data, the generated total delay is the following:

$$T_{tot} = \sum_{i=1}^n i \times (T_{hop-by-hop}) \quad (1 < n) \quad (8)$$

Then in equation (8) represents the total hop counts. It has more than 1 for the communication by the neighbor node.

We assume that the system and network parameter which are Encryption delay, Decryption delay, Delay for the allocation and channel access, Data transfer delay and Number of nodes in the entire network. Table 3 shows that the system and network parameter

Table 3. The system and network parameters

Encryption delay, t_{Enc}	449ms
Decryption delay, t_{Dec}	456ms
Delay for the allocation and channel access, Δt	10ms
Data transfer delay, $t_{Transmission}$	10ms
Number of nodes in the entire network, n	215

In Table3, Encryption delay and Decryption delay are result of encryption and Decryption in Table2 and Number of nodes in the entire network is 215 which is less than the maximum number of nodes 65,535 in the WPAN (Wireless Personal Area Network) [1] area. Delay for the allocation and channel access and Data transfer delay is 10ms.

Figure 14 show that total delay according to the count of hop between CH and N_i .

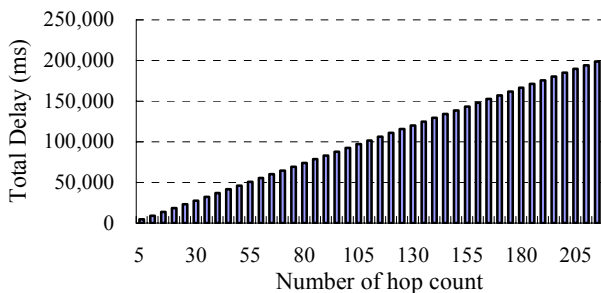


Figure 14. Total delay according as the count of hop

In figure 14, the delay of 30 hops and 180 hops generate 27,450ms, 164,700ms respectively. If the number of nodes in the entire network is 65,535 (the maximum number of nodes in the sensor networks [1]), the delay is measured 59,964,525ms (about 16 hours). The fundamental reason of the extensive delay occurred is the performance of the equipment that used in the experiment as 8-bit Microcontroller has a low capability of the operation.

Therefore, the scale of sensor networks consisted of the equipments increases, the transmission delay and energy consumption will also increases.

V. CONCLUSIONS

In this paper, we analyses the performance of AES encryption algorithm in the symmetric key encryption on ATmega644p in 8-bit microcontroller. In application scenario, we measure the encryption and decryption operation time by the plaintext size. As a result, scale of the sensor networks grows, the delay has been doubled. And energy consumption has also increased accordingly. In the future, specific researching on the performance analysis under plaintext size and hop count require.

ACKNOWLEDGMENT

This work was supported by the IT R&D program of MKE/IITA [2008-S-041-01, Development of Sensor Networks PHY/MAC for the u-City]

REFERENCE

- [1] IEEE Std 802.15.4: "Wireless Medium Access Control(MAC) and Physical Layer(PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)", 2003.
- [2] Yun Zhou, Yuguang Fang, Yanchao Zhang, "Securing wireless sensor networks: a survey," IEEE Communications Surveys and Tutorials, Vol. 10, No. 3, 3rd Quarter, 2008.
- [3] FIP 197: Announcing the Advanced Encryption Standard, Nov. 26., 2001. <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>.
- [4] J. Daemen and V. Rijmen, "AES Proposal: Rijndael, AES Algorithm," Submission, September 3, 1999.
- [5] M. Feldhofer, J. Wolkstorfer, and V. Rijmen, "AES implementation on a grain of sand," IEE Proc. Inf. Security, vol. 152, IEE, pp. 13-20, Oct. 2005.
- [6] RFC3602 The AES-CBC Cipher Algorithm and Its Use with IPsec. S. Frankel, R. Glenn, S. Kelly. September 2003
- [7] S. Kim, Ingrid Verbauwhede, "AES implementation on 8-bit microcontroller," Department of Electrical Engineering, University of California, Los Angeles, USA, September, 2002.
- [8] Atmel, 8-bit Microcontroller with 16/32/64K Bytes In-SystemProgrammable Flash., E ed., Atmel, San Jose, CA, 08 2008. http://www.atmel.com/dyn/resources/prod_documents/doc7674S.pdf.
- [9] A. Perrig et al., "SPINS: Security Protocols for Sensor Networks," ACM Wireless Networks, vol. 8, no. 5, Sept. 2002.