JANT 한국항행학회논문지
Journal of Advanced Navigation Technology

항공 우주 해상

J. Adv. Navig. Technol. 23(5): 339-344, Oct. 2019

암호화 AES Rijndael 알고리즘 적용 유도탄 점검 장비

Guided Missile Assembly Test Set using Encryption AES Rijndael Algorithm

정 의 재*· 고 상 훈 · 이 유 상 · 김 영 성
LIG넥스원(주) 유도무기연구소

Eui-Jae Jung* · Sang-Hoon Koh · You-Sang Lee · Young-Sung Kim
PGM Research and Development Lab, LIGNex1 Co., Gyeonggi-do 13488, Korea

[요 약]

정보통신 기술 발전에 따른 데이터 보안 위협의 상승에 대비하기 위하여 유도탄 점검 장비에 저장된 자료의 안전성을 보장할 수 있는 기술은 중요하다. 이를 위하여 자료가 누출 되더라도 복원할 수 없게 데이터 저장 시 암호화를 수행하여야 하고, 해당 데이터를 복호화한 후에도 무결성이 보장되어야 한다. 본 논문에서는 데이터 저장 시 대칭키 암호시스템인 AES 알고리즘을 유도탄 점검 장비에 적용하고, 각 AES의 각 비트 별 데이터 양에 따른 암호화 복호화 시간을 측정하였다. 또한 기존 점검 시스템에 AES Rijndael 알고리즘을 구현하여 암호화 수행으로 인한 영향을 분석하였고 제안한 암호화 알고리즘을 기존 시스템에 적용하는 것이 적합한지 확인 하였다. 용량별 / 알고리즘 비트수별로 분석한 결과 제안한 알고리즘 적용이 시스템 운용에 영향 없음을 확인하였고, 최적의 알고리즘을 도출할 수 있었다. 추가로 복호화 결과를 초기 데이터와 비교하였고, 해당 알고리즘이 데이터 무결성을 보장할 수 있음을 확인할 수 있었다.

[Abstract]

In order to prepare for the rise of data security threats caused by the information and communication technology, technology that can guarantee the stability of the data stored in the missile test set is important. For this purpose, encryption should be performed when data is stored so that it cannot be restored even if data is leaked, and integrity should be ensured even after decrypting the data. In this paper, we apply AES algorithm, which is a symmetric key cryptography system, to the missile test set, and Encrypt and decrypt according to the amount of data for each bit of each AES algorithm. We implemented the AES Rijndael algorithm in the existing inspection system to analyze the effect of encryption and apply the proposed encryption algorithm to the existing system. confirmation of suitability. analysis of capacity and Algorithm bits it is confirmed that the proposed algorithm will not affect the system operation and the optimal algorithm is derived. compared with the initial data, we can confirm that the algorithm can guarantee data undulation.

Key word : Missile assembly test set, Advanced encryption standard Rijndael algorithm, Encryption, Decryption.

https://doi.org/10.12673/jant.2019.23.5.339

Received 29 August 2019; **Revised** 4 October 2019
Accepted (Publication) 18 October 2019 (30 October 2019)

***Corresponding Author; Eui-Jae Jung**

Tel: +82-31-8026-7447
E-mail: euijae.jung@lignex1.com

Ⅰ. 서 론

오늘날 사회는 정보화 사회로서 많은 양의 정보들이 컴퓨터에 의해 처리 및 저장되고 있으며 각종 전산망을 통해 그 사용범위가 확대되고 있다. 이러한 시점에서 정보들을 안전하게 보관하고 전송하는 정보보호문제는 컴퓨터에 저장된 정보의 효율적인 사용과 전산망 활용의 극대화라는 측면에서 매우 중요한 문제로 인식되고 있다. 이러한 추세에 따라 방위산업 분야의 경우 유도탄 점검 장비에서 산출된 데이터 보안에 대해 더욱 강력히 요구되어 우리의 독자적인 암호알고리즘의 개발과 이에 대한 안전성을 평가하여 공인할 수 있는 분석 능력을 갖추는 것은 중요하다. 군사 목적의 다양한 산출 데이터를 다루는 유도탄 점검 장비에서 자료의 안전성과 무결성 보장을 위해 송수신 되는 데이터와 대용량 자료의 안전한 저장을 보장 할 수 있는 보안 기술 연구는 중요하다. 이러한 유도탄 점검 장비에서 산출된 데이터들은 다양한 경로를 통해 보안 취약점을 가질 수 있다. 이를 위해 자료가 누출 되더라도 원래의 데이터를 알아보지 못하게 하는 비밀성 보장과, 침입자가 기존자료에 틀린 자료를 삽입하거나 자신에게 유리한 자료로 대체 시키지 못하게 하는 인증성을 통한 무결정 보장으로 보안 취약점을 낮추도록 하여야 한다[1].

본 논문에서는 유도탄 점검 장비에서 산출된 데이터를 대칭키 암호화 AES(advanced encryption standard) Rijndael 알고리즘을 이용하여 보안 안정성 및 인증성이 적용된 효율적인 시스템 구조에 대해 제안하며, 구현한 AES Rijndael 알고리즘의 실행시간 분석 및 데이터 무결성 확인을 통해 향후 개발하는 점검 장비에의 적용에 대한 적합성을 분석 평가하였다.

Ⅱ. AES Rijndael 알고리즘 개요와 코드 구현

AES Rijndael 알고리즘의 개요와 구조를 기술하고 암호화 AES Rijndael 적용 유도탄 점검 장비를 제안하기 전 AES Rijndael 알고리즘을 프로그래밍으로 코드 구현 및 기술 하였다.

2-1 AES Rijndael 알고리즘

1) AES Rijndael 알고리즘 개요와 적용 이유

대칭 키 암호화 알고리즘의 종류 중 가장 보편적으로 쓰이는 암호화 방식은 현 미국 표준 방식은 AES Rijndael 알고리즘으로 암호화와 복호화 과정에서 동일한 키를 사용하는 대칭 키 알고리즘 이며 속도와 코드 효율성이 장점이다. AES Rijndael 알고리즘을 적용한 이유는 기존에 보편적으로 사용했던 DES(data encryption standard) 알고리즘의 보성성 취약에 대한 대체 요구로 신규 알고리즘에 대한 경쟁을 하게 되었고, 해당 기법이 최종 선정되었다[2]. DES알고리즘의 주요 논란의 대상이 되는 문제는 암호문에 56비트 키를 사용 시 컴퓨터의 기술의 급속한 발전에 따라 전수 키 탐색 공격(exhaustive key search attack) 방법이나 매개 변수로 상황을 달성하려는 일종의 암호화 공격(time memory trade off)에 쉽게 공격될 수 있다는 점이다. 이러한 문제를 해결하고 암호화된 키의 비밀성을 높이기 위하여 블록 암호 형식의 AES Rijndael 알고리즘을 미국 정부 표준 지정하였다. 최종적으로 AES Rijndael 알고리즘을 미국 표준 기술 연구소가 5년 표준화 과정을 거쳐 연방 정보 처리 표준으로 발표 하였다. 특징으로는 키의 크기(블록의 크기)로 128/ 192/ 256비트 등 32배수 확장 가능하며 확장된 길이에 따른 연산 횟수도 변경된다. 또한 3가지 종류의 비트길이에 따라 암호화가 복잡도가 높아진다[3].

2) AES Rijndael 알고리즘 분석

AES Rijndael 알고리즘은 대칭 블록 암호화 알고리즘이며, 4행 4열 바이트 행렬인 128/ 192/ 256 비트의 각각의 데이터 블록(state)으로 분할하여 각 블록에 대한 SubBytes, ShiftRows, MixCoulmns, AddRoundKey의 연산을 반복을 실행한다. SubBytes의 Sub는 substitution의 약자로써 데이터의 각 바이트 단위로 치환을 수행하고, ShitftRows는 각 행에 대하여 바이트 단위로 좌측으로 순환 시킨다. MixCoulmns는 열에 적용된 선형 변환으로 행렬 곱셉을 연산한다. AddRoundkey는 각 라운드 키를 배타적 논리합 연산을 수행한다. AES Rijndael 알고리즘 생성과정은 그림 1 과 같다. 평문과 최초 비밀 키 사이에 AddRoundkey에서 각 라운드를 위해 생성된 라운드 키를 이용하게 되고 마지막 n번째에 라운드 에서는 MixCoulmns가 수행되지 않으며 복호화는 암호화의 역순으로 진행된다[4][5].

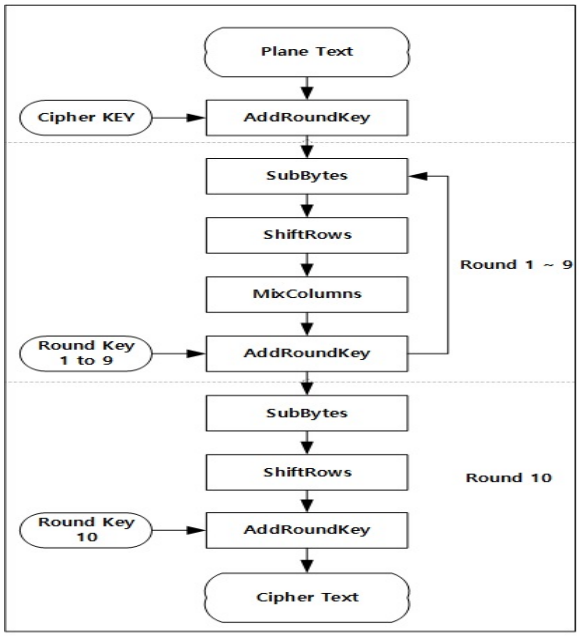


그림 1. AES Rijndael 알고리즘 흐름도 [AES-128]
Fig. 1. Flow chart of AES Rijndael algorithm.

표 1. 블록 길이와 키 길이에 따른 라운드 수
Table 1. Number of rounds according to block length and key length.

AES Rijndael algorithm	Key length (N_k words)	Block length (N_b words)	Round key count (N_r counts)
AES-128	4 words	4 words	10 counts
AES-192	6 words	4 words	12 counts
AES-256	8 words	4 words	14 counts

표 1 과 같이 Round key 반복횟수는 AES Rijndael 알고리즘의 적용 길이에 따라 달라진다. 키 길이 N_k와, 라운드 수 N_r은 AES Rijndael 적용에 따라 길이가 달라지며, 블록크기 N_b는 4워드로 고정된다.

3) AES Rijndael 알고리즘 키 확장

AES Rijndael 128/ 192/ 256 비트 각각의 알고리즘은 평문을 암호화하기 위한 과정에서 각 라운드에 사용하는 라운드 키를 생성하기 위해 먼저 키 확장(key expansion)과정을 진행한다. 생성된 라운드 키는 암호화 전에 평문과 배타적논리합으로 연산하고 라운드 키들은 라운드 마지막 단계에서 AddRoundKey 연산을 한다. 키 확장은 연산방법은 각각의 비트를 4개의 워드로 바꾼 후 마지막 워드는 왼쪽으로 이동(left rotation)한다. 이동이 완료된 워드는 S-box 연산을 수행한 후 라운드 워드가 생성 될 때 배타적 논리합으로 연산한다.

그림 2 와 같이 키 확장 방식을 의사코드로 구현할 수 있다. temp 변수는 현재의 블록이 아닌 직전의 블록에 속하는 워드 값을 임시로 저장하기 위한 것이며, key 변수는 워드 내의 각 바이트 값을 뜻하고 mod 연산은 나머지 연산을 뜻한다. 128비트 키일 경우 i는 0 ~ 43의, 192비트 0~51, 256비트 0~59의 범위를 가진다. 256비트일 경우는 128/ 192비트와 차이점이 있다. 256비트일 경우 변수 i를 Nk 값으로 나눈 나머지가 4일 경우에는 Rotword 연산과 라운드 상수 RCon과의 배타적 논리합 연산이 생략되고 워드의 각 바이트를 S-box로 이용하여 대치한 워드 단위 추출 함수인 Subword만 수행된다. 그림2 는 키 확장 과정을 의사코드로 나타내었다[6].

```
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
        end while
end
```

그림 2. 키 확장과정의 암호화 의사코드
Fig. 2. Key encryption pseudocode of extension process.

```
for i = 0 step 1 to (Nr + 1) * Nb – 1
        dw[i] = w[i]
End for

for round = 1 step 1 to Nr – 1
        InvMixColumns(dw[round *m Nb, (round+1) * Nb - 1])
End for
```

그림 3. 키 확장과정의 복호화 의사코드
Fig. 3. Key description pseudocode of extension process.

AES Rijndael 알고리즘 키 확장 과정에서 복호화 과정은 키 공급 순서가 역순으로 발생한다. 그림 2 와 동일하게 암호화 의사코드를 동일하게 사용하고 그림3 과 같이 코드를 키 확장 과정의 의사코드에 마지막에 추가하면 암호화와 동일한 연산 순서를 유지하면서 복호화를 수행할 수 있게 된다.

2-2 AES Rijndael 알고리즘 프로그래밍 코드 구현

1) AES Rijndael 알고리즘 코드 구현

AES Rijndael 알고리즘을 프로그램 코드로 구현하기 위해서는 필수적으로 AES Rijndael 알고리즘이 적용 된 클래스와 관련 함수가 필요하다. AES Rijndael 알고리즘 구현하기 위해서는 API(application programming interface)에서 제공된 AES Rijndael 암호화 알고리즘 클래스와 관련 함수로 AES Rijndael 알고리즘을 구현하였다. 위에서 기술된 알고리즘 분석을 AES Rijndael 알고리즘 코드로 구현 시, 제공된 API 클래스와 함수는 미사용 시 시스템 설계는 복잡도가 상승한다. 표 2 는 AES Rijndael 알고리즘을 코드구현에 필요한 주요 클래스, 함수, 속성, 인터페이스이며, 표 2 의 클래스와 주요 함수를 이용하여 그림 4 의 AES Rijndael 알고리즘을 코드로 구현하였다.

표 2. AES Rijndael 알고리즘 클래스 및 주요 함수
Table 2. Number of rounds according to block length and key length.

Class	
using	System.Security.Cryptography
Aes()	Represents the abstract base class from which all implementations of the advanced encryption standard(AES) must inherit
Methods	
Create()	Creates a cryptographic object that is sed to perform the symmetric algorithm
CreateEncryptor()	Creates a symmetric encryption object with the current Key.
Properties	
KeySize()	Gets or sets the size in bits
Key()	Gets or sets the secret key for the symmetric algorithm
IV()	gets or sets the initialization vector for symmetric algorithm
Interface	
ICryptotransform()	Defines the basic operations of cryptographic transformations

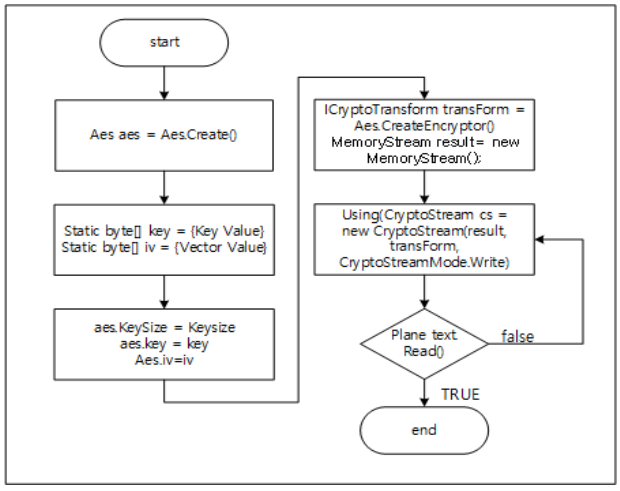


그림 4. AES Rijndael 알고리즘 코드 흐름도
Fig. 4. Flow chart of AES Rijndael algorithm code.

Ⅱ. AES Rijndael 알고리즘 적용 유도탄 점검 장비 암호화 시스템

1) 유도탄 점검 장비 암호화 시스템 구축

유도탄 점검 장비는 유도탄의 성능을 보장하는 장비로, 유도탄 성능에 대한 결과가 데이터로 저장된다. 유도탄 점검 장비 암호화 시스템 구축 시 점검 장비의 모의명령 송신 데이터, 대상 유도탄의 전자장치 송신에 대한 모의명령 응답 데이터와 유도탄 내부 상태를 나타내는 실시간 상태 데이터를 대상으로 수행해야 한다. 그림 5 의 기존 유도탄 점검 장비는 유도탄 점검 장비와 유도탄 내부 전자장치와 데이터를 모의명령 데이터, 모의 응답데이터, 실시간 상태 데이터를 저장하였다. 저장 시 텍스트 파일을 저장하거나, DBMS(data base management system)에 데이터를 저장하였다. 기존 장비의 문제는 자료가 누출되더라도 원래의 데이터를 알아보지 못하게 하는 비밀성과 침입자가 기존자료에 틀린 자료를 삽입하거나 자신에게 유리한 자료로 대체 시키지 못하게 하는 인증성이 낮다는 것이다. 또한 저장된 데이터에는 보안사항에 해당하는 값이 일부 포함되어 있으므로 보안에 유의하여야 한다. 이를 위해 데이터의 인증성 및 무결성이 보장되어야 한다. 이를 위해 AES Rijndael 알고리즘과 같은 암호화 알고리즘을 적용할 필요가 있다.

그림 6 와 같이 AES Rijndael 알고리즘이 적용 된 유도탄 점검 장비를 구축하면 데이터 암호화 절차를 수행하여 비밀성과 인증성을 높이고 데이터 보안 취약점 낮출 수 있다. 유도탄 점검 장비는 유도탄 점검 장비의 프로세스 내부에 암호화 키(cipher key)를 저장하고, DBMS에 저장하려는 송신데이터, 모의 명령 응답 데이터, 실시간 상태 데이터를 암호화 하여 저장하고, 데이터 분석 시 암호화 키를 역순으로 복호화 하여 데이터를 전시하고, 데이터를 산출 할 수 있어야 한다. 유도탄 점검 장비를 구축하기 전 선행되어야 하는 부분은 DBMS에 대한 구축, 운영체제 커널 내 암호화 키의 메모리 확보이다. DBMS의 데이터베이스는 데이터 필드 확보를 수행하여야 하며, 유도탄 점검 장비와 통신이 가능한 상태이어야 한다.

AES Rijndael 알고리즘을 적용한 유도탄 점검 장치에서는 유도탄 점검 장비가 통신모듈을 통해 데이터를 대상 유도탄의 전자장치에 요청하고 응답 데이터와, 실시간 데이터를 유도탄 점검 장비 내부 암호화 키와 AES Rijndael 알고리즘 코드 연산을 통해 암호화된 데이터를 DBMS에 저장한다. 복호화 시 DBMS에 존재하는 데이터를 암호화 키와 AES Rijndael 알고리즘 키를 이용해 역산하여 전시한다. 관리소홀로 인한 암호화 키의 유실을 방지하기 위해여 암호화 키를 운영체제 내부 커널에 저장을 하였다.

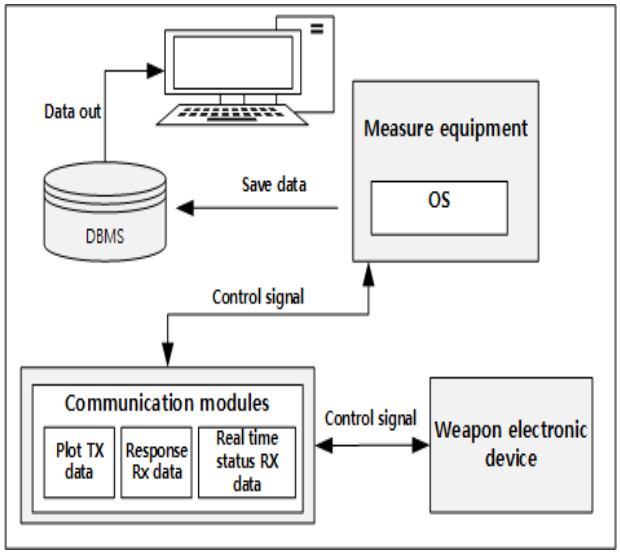


그림 5. 기존 유도탄 점검 장비 시스템
Fig. 5. Before missile test set.

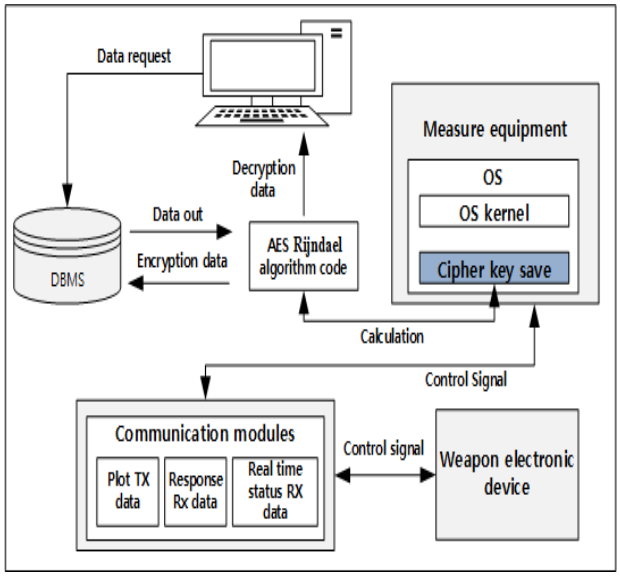


그림 6. AES Rijndael 알고리즘 적용 유도탄 점검 장비
Fig. 6. Missile assembly test set using encryption AES Rijndael algorithm.

2) AES Rijndael 알고리즘 적용 유도탄 점검 장비 성능분석

본 논문에서 제안한 암호화 AES Rijndael 알고리즘을 유도탄 점검 장비에 맞게 구현하였고, 데이터 암호화 및 복호화 알고리즘 실행 후 처리속도를 시험하였다. 대상이 되는 평문과 AES Rijndael 알고리즘을 운영체제 커널 내부 암호화키와 함께 연산하여 AES Rijndael 알고리즘 비트길이와 데이터의 양에 따른 암호화 복호화 시간을 측정하였으며, 유도탄 점검 장비에 AES Rijndael 알고리즘 적용 시 적합성을 수치로 나타내고 적용가능 여부를 확인하였다. 분석을 위한 시스템 규격은 Intel core I7 4.20 GHz, 32.0 GByte RAM, Windows 10 OS 환경으로 구성하였다. 성능분석의 방법은 실시간 상태 데이터를 파일형태로 저장 후 AES Rijndael 알고리즘 키 128/ 196/ 256 비트 별 AddRoundKey횟수와 키 확장 과정을 수행하였고, 파일용량은 점검 장비 시스템에서 운용하는 1 Mbytes와 향후 확장성을 고려하여 5 Mbytes및 10 Mbytes를 대상으로 동일한 조건 하에 수행하였다. 또한 암호화 수행 시 데이터는 동일 데이터를 각 AES Rijndael 비트 별 10회 수행 결과 및 2-2 AES Rijndael 알고리즘 프로그램 코드 구현에 따라 함수를 적용하여 실행시간을 측정하였다. 그림 7 ~ 9는 각각 128/ 192/ 256비트 별 AES Rijndael 블록길이별로 측정한 결과이다.

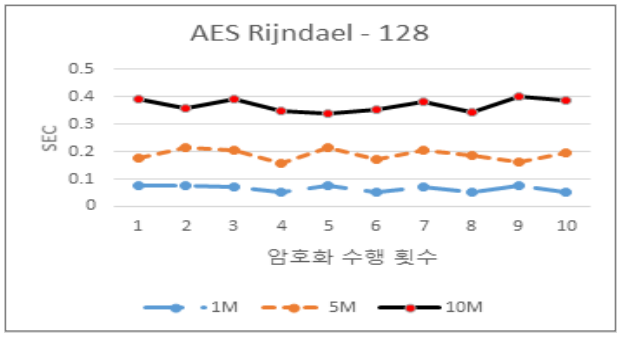


그림 7. AES Rijndael – 128 비트 연산 수행 시간 측정
Fig. 7. AES Rijndael – 128 bit measure operation execution time.

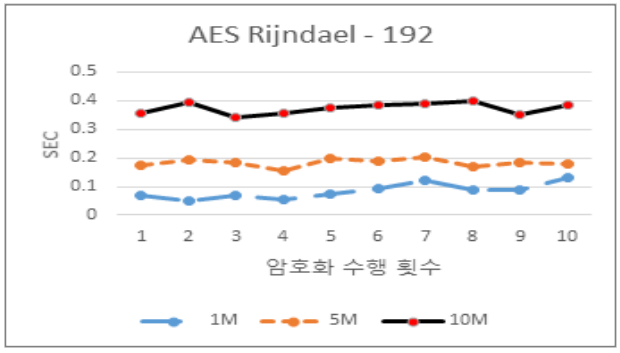


그림 8. AES Rijndael – 192 비트 연산 수행 시간 측정
Fig. 8. AES Rijndael – 192 bit measure operation execution time.

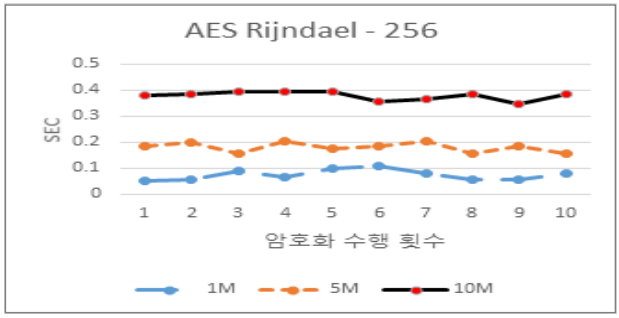


그림 9. AES Rijndael – 256 비트 연산 수행 시간 측정
Fig. 9. AES Rijndael – 256 bit measure operation execution time.

데이터의 길이를 1 Mbytes, 5 Mbytes, 10 Mbytes로 설정 후 AES Rijndael 알고리즘 적용한 결과 30회의 평균 처리 속도가 0.065 sec, 0.187 sec, 0.37 sec로 측정되었다. 점검 별 대기시간을 고려하면 충분히 짧은 시간으로 시스템 운용에 영향을 주지 않는다. 또한, 라운드 키 길이 변화에 따른 처리속도 차이는 0.01 sec로 AES Rijndael 128/ 192/ 256 비트 별 차이는 평문 데이터 길이에 대한 차이인 0.305 sec 보다 현저히 낮아 처리속도에 대한 비트별 차이는 고려하지 않아도 된다. 연산비트가 클수록 Round key 횟수가 증가하여 암호화의 안전성은 높아지는데, AES Rijndael 알고리즘의 각 비트별 처리속도 차이를 고려하지 않아도 되기에 시스템에 256비트 AES Rijndael 알고리즘을 적용하는 것이 타당하다. 또 다른 결과로는 1/ 5/ 10 Mbytes 데이터 길이에 처리속도 차이가 128 / 192 / 256 비트 알고리즘에 따른 차이 보다 더 큰 것을 확인 하였는데, 이러한 차이는 평문 데이터를 암호화 하는 시간보다 메모리에서 처리하는 시간이 암호화 연산시간에 더 큰 영향을 주기 때문에 파일 크기별 시간차가 더 큰 것 이다. 암호화 된 각 경우의 데이터를 복호 화한 결과, 평문 데이터와 일치함을 확인하였다.

Ⅲ. 결 론

본 논문에서는 암호화 AES Rijndael 알고리즘을 유도탄 점검 장비 적용하고 각 통신모듈에서 생성된 데이터를 각 1/ 5/ 10 Mbytes로 저장하여 AES Rijndael 알고리즘 적용 후 실시간 처리 속도를 시간으로 측정하였다. 평문 데이터가 커지면서 암호화 수행 시간을 확인하였고, 암호화 수행시간 보다는 평문 데이터를 메모리 처리 과정 시간이 암호화 수행시간 보다 영양을 준다는 것을 확인하였다. 그리하여 점검 장비에 AES Rijndael 알고리즘을 적용하여 데이터 안전성을 보장할 수 있음을 확인하였고, 데이터 복호화 후 평문데이터와의 비교를 통하여 무결성이 보장됨을 확인하였다.

AES Rijndael 알고리즘 적용 시 유도탄 점검 장비에서 평문 데이터와 힙 메모리 연산속도를 개선을 위한 적합한 하드웨어 구현과 암호화 결과의 견고성에 대한 정량적인 분석결과를 도출하는 실험이 진행되어야한다.

References

[1] E. Biham and A. Shamir, "Differential cryptanalysis of DES-like cryptosystem," *Journal of cryptology*, Vol.4, No.1, 1991.

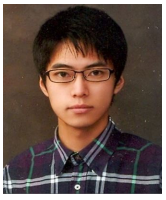
[2] M. S Hwang, "An elgamal-like cryptosystem for enciphering large messages," *IEEE Transactional on Knowledge and Data Engineering*, Vol. 14, No. 2, pp. 445-446, 2002.

[3] M. N. Islam , M. Mia, M. Chowdhury, and M. A. Matin, "Effect of security increment to symmetric data encryption through AES methodology," in *Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*, Phuket: Thailand, pp. 291-294, 2008.

[4] B. Y. Choi, "Design of AES Rijndael cryptographic processor," *Journal of Korean Institute of Communication Sciences*, Vol. 26, No. 10B, pp. 1491-1500, Oct. 2001.

[5] J. Y Oh and Sun J. H Sun, " Experimental analysis of the AES encryption algorithm", *Journal of Korea Institute of Information, Electronics, and Communication Technology*, Vol. 3, No. 2, Jun. 2010.

[6] S. M. Kim, *Cryptographic Algorithms Understand by Story,* Seoul, Korea: Road Book, pp. 197-224, 2017



정 의 재 (Eui-Jae Jung)
2008년 2월 : 충주대학교 컴퓨터공학과 (공학사)
2008년 2월 ~ 2016년 2월 : ACE Technology S.W연구소 선임연구원
2016년 2월 ~ 현재 : LIG넥스원 유도무기연구소 선임연구원
※ 관심분야 : 계측제어, 정보통신, 유도무기

고 상 훈 (Sang-Hoon Koh)
2011년 2월 : 고려대학교 전기전자컴퓨터공학과 (공학석사)
2011년 1월 ~ 현재 : LIG넥스원 유도무기연구소 선임연구원
※ 관심분야 : 유도무기, 정보통신

이 유 상 (You-Sang Lee)
2006년 2월 : 경희대학교 컴퓨터공학과 (공학사)
2006년 3월 ~ 현재 : LIG넥스원 유도무기연구소 선임연구원
※ 관심분야 : 유도무기, 정보통신

김 영 성 (Young-sung Kim)
2017년 2월 : 한양대학교 컴퓨터공학과 (공학석사)
2017년 2월 ~ 현재 : LIG넥스원 유도무기연구소 선임연구원
※ 관심분야 : 유도무기, 무인항공기 지상체, 자율주행, PUF