AES 암호화 알고리즘의 실험적 분석

오 주 영*, 서 진 형**

Experimental Analysis of the AES Encryption Algorithm

Ju-Young Oh*, Jin-Hyung Suh**

요 약

암호화 작업은 많은 연산시간을 요한다. 본 논문에서는 암호화에 수반되는 연산시간을 분석하기 위해 평문의 압축과 가변의 블록 크기, 라운드 횟수의 사용자 설정 및 단계별 작업과정의 선택적 적용 등의 네 가지 원리에 기초해서 AES 구조를 확장하였다. 실험은 C++로 수행하였으며 실험을 통해 최적의 입력 블록 크기를 도출하였다.

ABSTRACT

Cryptography is primarily a computationally intensive process. In this paper we expand AES scheme for analysis of computation time with four criteria, first is the compression of plain data, second is the variable size of block, third is the selectable round, fourth is the selective function of whole routine. We have tested our encryption scheme by c++ using MinGW GCC. Through extensive experimentations of our scheme we found that the optimal block size.

Keywords : Encryption, AES(Advanced Encryption Standard), S-Box, Round Key,MinGW GCC

Ⅰ. 서 론

암호화는 최근의 정보 유통에 있어서 의도적 편취나 정보변경을 차단하기 위한 필수적인 보안절차가 되었다[1]. 최근의 관련 연구는 암호화의 소재별 성능이나 암호화 자료의 안전성에 기초한 연구가 진행되고 있다[2]. 특히, 암호화에 수반되는 실행 시간과 자원사용량을 무선 환경의 이동 단말기에 적합한 수준으로 최소화하기 위한 다양한 하드웨어 구현 방법과 성능 분석에 대해 연구되고 있다[3].

블록 암호화 기법인 AES의 경우 구현상의 복잡도는 블록의 크기에 비례하여 커지므로 너무 크게 설정하면 효율성이 떨어지게 되고, 너무 작게 설정하면 여러 유형의 암호 공격으로부터 안전할 수 없게 된다[4-5]. 뿐만 아니라, 암호문의 견고성은 라운드 횟수를 늘리고 입력키를 크게 설정함으로써 향상시킬 수 있다[5].

본 논문에서는 AES 암호 알고리즘을 구성하는 알고리즘 단계를 유연하게 선택적으로 적용가능하게 하고, 알고리즘 효율의 최대변수가 되는 데이터 블록과 키 블록의 크기를 가변으로 적용할 수 있도록 하였으며, 라운드 횟수를 사용자 지정한도만큼 반복할 수 있도록 확장하여, 알고리즘의 주요

*교신저자 경인여자대학 정보미디어학부 (odid080@kic.ac.kr)
**경인여자대학 정보미디어학부 (jhsuh@kic.ac.kr)
접수일자 : 2010년 5월 4일, 수정일자 : 2010년 5월 19일, 심사완료일자 : 2010년 6월 11일

변수에 대해 실행시간을 벤치마킹함으로써 효율적 구현을 위한 실험도구로 활용할 수 있도록 하였다. Ⅱ장에서는 AES 확장 알고리즘의 작업과정을, Ⅲ장에서는 실험 및 결과를 Ⅳ장에서 결론을 기술하였다.

Ⅱ. AES

1. AES-Rijndael

AES-Rijndael은 대칭 블록 암호화 알고리즘이며 SubBytes, ShiftRow, MixColumns 및 AddRoundKey의 연산을 4행 4열의 바이트 행렬인 128 비트의 데이터 블록(State)으로 분할하여 각 블록에 대해 반복 실행함으로써 암호문을 생성하며 과정은 그림 1과 같다. 암호화 연산은 평문과 최초 비밀 키 사이에 AddRoundKey가 이루어진 다음에, 네 가지 변환들이 순서대로 처음 (n-1)번의 라운드에 대하여 동일하게 반복된다. AddRoundKey에서는 각 라운드를 위해 생성된 라운드 키를 이용하게 되고 마지막 n번째 라운드에서는 MixColumns가 수행되지 않으며 복호화는 암호화 과정의 역순으로 진행된다[6].

```
State = X
AddRoundKey(State, Key0)
    for r = 1 to Nr - 1
        SubBytes(State, S-box)
        ShiftRows(State)
        MixColumns(State)
        AddRoundKey(State, KeyR)
    endfor
SubBytes(State, SBox)
ShiftRows(State)
AddRoundKey(State, KeyNr)
Y = State
```

그림 1. AES 암호화 알고리즘
Fig. 1. AES encryption algorithm

2. AES 확장 모델

AES-128/192/256 각각의 알고리즘은 평문을 암호화하기 위한 각 알고리즘의 작업과정에서 알고리즘의 키 블록 크기와 라운드 횟수에 따른 분류명칭이다. 블록 암호화 과정의 자원 점유와 상대적 실행효율의 최대 가늠자는 식 (1)과 같이 데이터 블록의 크기와 암호화 진행을 위한 라운드 횟수에 의해 결정된다[3].

$$Tp = \frac{BlockSize}{(rn+1) \times Clockcycle} \quad (1)$$

rn은 라운드 횟수, $Blocksize$는 각 평문 블록의 입력 크기를 명시하는 변수이므로 암호화 처리율은 입력 데이터 블록의 크기와 암호화 진행을 위한 라운드 횟수에 의해 좌우된다. 반면, 외부공격에 대한 암호문의 견고성은 라운드 횟수에 비례해서 증가하게 된다.

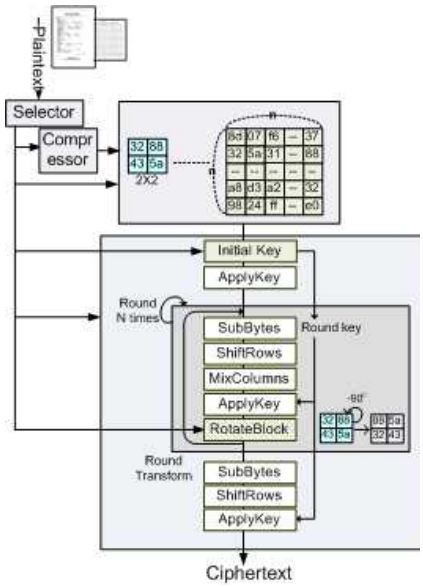

그림 2. 확장된 AES 구조
Fig. 2. Extended AES Scheme

하지만, 암호화를 진행하는 시스템의 가용자원에 따라 입력 평문 블록의 크기와 라운드 횟수의 수용범위는 달라질 것이다. 따라서, 암호화를 진행하

는 목적 시스템에서 암호화를 위한 주요 변수에 따른 실행시간 척도의 역할 도구가 요구되므로 구현된 AES 모델을 활용함으로써 이동형 단말기로부터 중형서버에 이르기까지 각기의 목적시스템에 적합한 제약사항을 만족시키기 위해 요구되는 블록화 계수와 알고리즘 라운드 수를 선택하는 가늠자로 활용할 수 있다.

암호화 대상 시스템에 맞는 다양한 설계 변수들을 설계자가 선별적으로 선택할 수 있도록 하는 목적함수를 만족시키기 위해 제안 모델에서 확장된 부분은, 입력 블록의 선택적 지정(2×2(32비트)~N×N), 암호화 연산의 선택적 삭제 또는 신규 함수의 적용, 라운드 횟수의 선택적 적용 등이며, 진행과정은 그림 2와 같다.

알고리즘 입력 전단에서 평문 데이터 블록의 크기를 결정하고, 데이터 블록에 대해 사용자 입력 암호화키와 applyKey 작업 과정 진행 결과에 대해 subBytes 작업을 진행한다. 이후, ShiftRows 과정을 n번째 행(n-1번 왼쪽 쉬프트)까지 수행하며, 현재까지의 암호화블록에 대해 키 스케줄에 의해 만들어진 키와 XOR 연산 이후, MixColumns 작업을 진행한다. RotateBlock 연산은 선험적으로 적용된 사용자 정의 함수이며 현재까지의 암호화블록 을 반시계방향으로 90도 회전한 후 한 번의 라운드를 종료하게 된다.

Ⅲ. 실험 및 결과

확장 모델의 구현은 Intel 2.4Ghz CPU와 2G RAM의 플랫폼에서 Code::Blocks C++로 구현하여 MinGW GCC로 컴파일 하였으며, 실험은 일반 문자조합으로 구성된 단순 텍스트 파일과, 특수문자와 그래프가 추가된 혼합파일 각 20종에 대해 진행하였으며 실험 데이터는 표 1과 같다.

먼저, 목적 시스템의 가용자원 내에서 최적 블록크기를 찾기 위해 10라운드 256비트 키에 대해 입력 블록의 크기를 2×2에서 900×900 범위에서 실험한 결과, 입력 블록의 크기에 의해 실행시간이 큰 편차를 보이는데, 그림 3의 1M 평문과 그림 4의 14M 평문 입력에 대해 암호화에 수반되는 최소 실행시간은 30×30 블록 크기에서 형성됨을 확인하였다.

표 1. 입력데이터
Table 1. Input data

	Data(KB)			Data(KB)	
	simple	complex		simple	complex
1	7	7	11	799	799
2	17	17	12	881	881
3	25	25	13	947	947
4	62	62	14	1136	1136
5	135	135	15	1550	1550
6	141	141	16	1693	1693
7	304	304	17	2443	2443
8	383	383	18	3480	3480
9	497	497	19	7602	7602
10	516	516	20	14401	14401

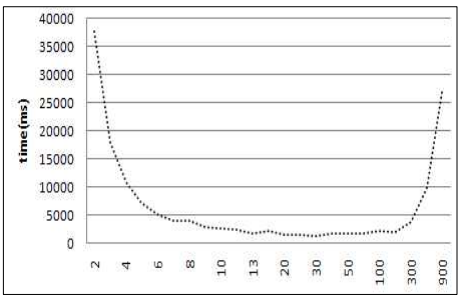

그림 3. 1M 파일 암호화 시간
Fig. 3. Encryption time of 1M file

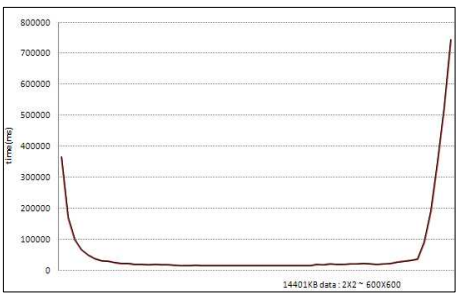

그림 4. 14M 파일 암호화 시간
Fig. 4. Encryption time of 14M file

그림 5와 그림 6은 단순파일과 혼합파일 각각에 대해 표 1의 각 20개 실험데이터에 대해 내부 라운드 10회와 암호화 키 160비트 크기에 대해 실험한 결과이며 입력 파일의 유형과 입력키에 의한 영향은 상대적으로 적으며, 라운드 횟수에는 비례하여 증가되는 결과를 보였으며 이러한 결과는 그

림 7과 그림 8의 라운드 횟수 14회와 암호화 키 256비트 크기에 대한 실험에서도 동일하였다.

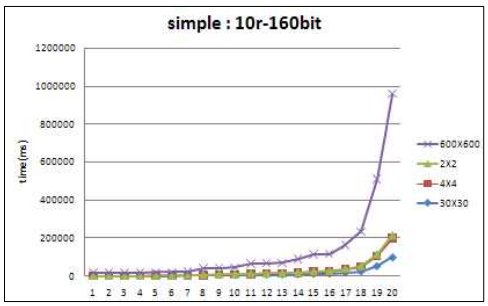

그림 5. 단순 파일 암호화 실행시간
Fig. 5. Encryption time of simple file

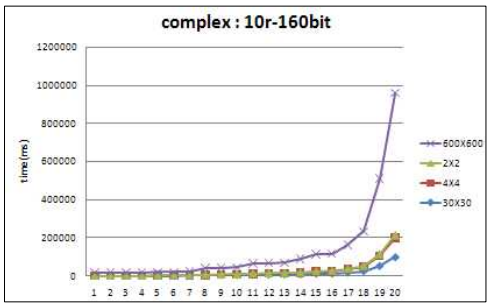

그림 6. 혼합 파일 암호화 실행시간
Fig. 6. Encryption time of complex file

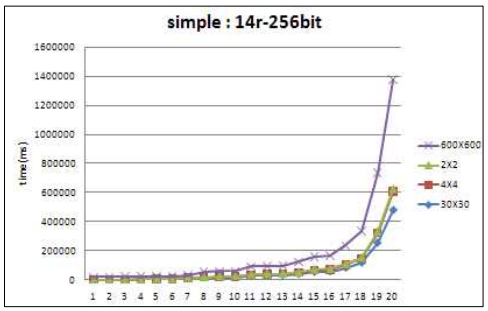

그림 7. 단순 파일 암호화 실행시간
Fig. 7. Encryption time of simple file

실험이 진행된 시스템 하에서의 암호화 실행시간은 명시된 실험 결과와 같이 진행된 모든 모형에서 평문의 입력 블록 크기 30×30에서 최적 결과를 형성하였고, 반복되는 라운드 내부 연산에 의해 크게 좌우됨을 확인하였다.

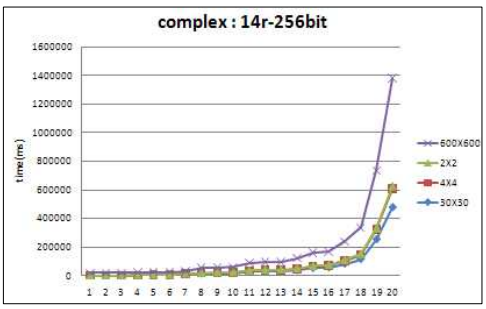

그림 8. 혼합 파일 암호화 실행시간
Fig. 8. Encryption time of complex file

따라서, 암호화가 필요한 대용량 자료 전송의 경우에는 입력 평문의 크기를 줄임으로써 입력 블록들에 의해 반복되는 라운드 수를 최소화해야 한다. 암호화 연산을 반복 수행하는 라운드는 입력 데이터 블록의 개수만큼 반복 실행되므로 블록의 입력 횟수를 줄여서 결과적으로 총 라운드 수를 줄일 수 있다. 이를 위해 입력 전단의 평문을 압축하여 암호화함으로써 라운드 수를 줄였다. 그림 9에서 압축 소요 시간을 포함하는 암호화 효율에서 압축 후 암호화 시간(AC)이 현격한 효율을 보임을 확인할 수 있다.

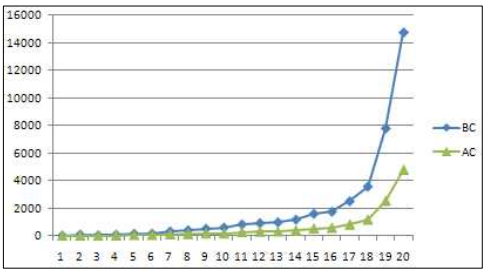

그림 9. 압축 전후의 암호화 실행시간
Fig. 9. Processing time of encryption before and after compression

IV. 결 론

본 논문에서는 암호화가 진행되는 다양한 목적 시스템의 플랫폼을 고려하여 자료 유형과 크기에 적합한 암호화 변수들인 평문 블록의 크기, 키 크

기, 알고리즘 내부의 라운드 횟수 등을 사용자 요구와 실행 자원의 수용범위에 맞게 선택적으로 설정하여 암호화 할 수 있도록 AES 구조를 확장하였다. 실험결과, 진행된 플랫폼에서의 최적의 암호화 계수로 30×30의 데이터 입력 블록 크기를 찾을 수 있었다. 뿐만 아니라, 라운드 횟수에 크게 종속되는 실행시간을 경감하기 위한 입력 전단의 평문을 압축하여 대형 자료의 보안 전송에 있어서 현격한 효율을 얻을 수 있음을 확인하였다.

본 실험을 기저로, 향후에는 라운드 연산에서 가장 큰 시간을 소모하는 SBox 최적 설계 기법과 라운드 연산속도 개선을 위한 파이프라인 삽입 및 이동형 단말기 등에 적합한 소규모 내장형 하드웨어 구현과 암호화 결과의 견고성에 대한 정량적인 분석 결과를 도출하는 실험이 진행되어야 한다.

참 고 문 헌

[1] Paul A. J, Paul Varghese, Mythili P, “A fast and secure encryption algorithm for message communication,” Information and Communication Technology in Electrical Sciences, pp. 629-634, 2007.

[2] Hua Li, Jianzhou Li, “A new compact dual-core architecture for AES encryption and decryption,” Electrical and Computer Engineering, Canadian Journal of Vol. 33, pp. 209-213, 2008.

[3] Gogniat G, Wolf T, Burleson W, Diguet J.-P, Bossuet L, Vaslin R, “Reconfigurable Hardware for High-Security/High-Performance Embedded Systems: The SAFES Perspective,” Very Large Scale Integration (VLSI) Systems, IEEE Transactions on 16(2), pp. 144-155, 2008.

[4] Alam, M, Ghosh S, Chowdhury D.R, Sengupta I, “Single Chip Encryptor/Decryptor Core Implementation of AES Algorithm,” 21st International Conference on VLSI Design, pp. 693-698, Jan. 2008.

[5] Lopez-Ongil C, Jimenez-Horas A, Portela-Garcia M, Garcia-Valderas M, Millan E.S, Entrena L, “Smart Hardening for Round-based Encryption Algorithms: Application to Advanced Encryption Standard,” On-Line Testing Symposium, pp. 167-168, 2008.

[6] Islam M.N, Mia M, Chowdhury M, Matin M.A, “ Effect of Security Increment to Symmetric Data Encryption through AES Methodology,” Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, pp. 291-294, 2008.

[7] Kuan Jen Lin, Chin-Mu Hiao, Ching Hung Jhan, “Exploring HW/SW Codesign of AES Algorithm Using Custom Instructions,” The 13th International Symposium on Consumer Electronics, pp. 192-195, 2009

저자약력

오 주 영(Ju-young Oh) **정회원**

1998년 2월 : 홍익대학교 전자계산과(이학 석사)
2004년 8월 : 홍익대학교 전자계산과(이학 박사)
2003년 3월 ~현재 경인여대 정보미디어학부교수

<관심분야> 설계자동화, 통합설계, 암호화/보안

서 진 형(Jin-Hyung Suh) **정회원**

1988년 2월 : 홍익대학교 전자계산과(이학 석사)
2010년 3월 : 한세대학교 정보통신공학(공학 박사)
1998년 3월 ~현재 경인여대 정보미디어학부교수

<관심분야> 멀티미디어, 운영체제, 보안