Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.26, NO.6, Dec. 2016 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2016.26.6.1353

AES/LEA 기반 이중블록길이 해쉬함수에 대한 효율성 분석*

김 도 원,[1†] 김 종 성[1,2‡]
[1]국민대학교 금융정보보안학과, [2]국민대학교 수학과

Analysis of the Efficiency for Some Selected Double-Block-Length Hash Functions Based on AES/LEA*

Dowon Kim,[1†] Jongsung Kim[1,2‡]
[1]Dept. of Financial Information Security, Kookmin University,
[2]Dept. of Mathematics, Kookmin University

요 약

본 논문에서는 블록암호를 기반으로 하는 대표적인 이중블록길이 해쉬함수 Abreast-DM, HIROSE, MDC-2, MJH, MJH-Double에 미연방표준암호 AES와 국내경량암호 LEA를 삽입하였을 때, 각각에 대한 효율성을 비교 분석하였다. AES는 공개된 최적화 소스코드를, LEA는 자체 구현한 소스코드를 이용하였다. 그 결과, 일반적으로 LEA 기반 해쉬함수가 AES 기반 해쉬함수보다 더 효율적이었다. 속도 면에서, Abreast-DM을 제외한 모든 해쉬함수에서 LEA가 AES보다 6%~19% 정도 더 빨랐다. 메모리 면에서도 AES의 고속구현 테이블로 인해 LEA가 20~30배의 효율성을 가졌다.

ABSTRACT

We analyze the efficiency of the double-block-length hash functions, Abreast-DM, HIROSE, MDC-2, MJH, MJH-Double based on AES or LEA. We use optimized open-source code for AES, and our implemented source code for LEA. As a result, the hash functions based on LEA are generally more efficient than those, based on AES. In terms of speed, the hash function with LEA are 6%~19% faster than those with AES except for Abreast-DM. In terms of memory, the hash functions with LEA has 20~30 times more efficient than those with AES.

Keywords: Hash function, Block cipher based Hash, Double-Block-Length, AES, LEA, Speed/Memory efficiency

I. 서 론

해쉬함수는 임의의 길이의 메시지를 입력으로 받아 이를 고정된 길이의 이진 수열로 변환하는 알고리즘이다. 어떤 비밀 정보도 사용하지 않는 공개 알고리즘이지만, 공개키 암호, 전자 서명, 패스워드 암호화, 키 공유, 의사난수발생기 등 암호 프로토콜 전반에 걸쳐 널리 쓰이는 기본적인 프리미티브이다. 현재까지 MD-4/5, SHA-0/1/2, HAVAL, RIPEMD-160, Whirlpool등 다양한 전용 해쉬함수가 제안되었지만, 2005년 Wang의 충돌쌍 공격 이후 다수의 해쉬함수에 취약성이 발견되었다[1,2].

Wang의 충돌쌍 공격이 발표된 이후, 해쉬함수의 설계와 분석에 관련한 방대한 연구가 이루어지기 시

Received(04. 06. 2016), Modified(1st: 08. 18. 2016, 2nd: 10. 18. 2016), Accepted(10. 18. 2016).
* 이 논문은 2016년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업 임(No. 2016R1D1A1A09919726).
† 주저자, kimdw0920@kookmin.ac.kr
‡ 교신저자, jskim@kookmin.ac.kr(Corresponding author)

작했다. 가장 널리 사용되는 SHA계열 해쉬함수를 제정했던 NIST는 2007년 새로운 표준 해쉬함수 공모 사업을 시작하였다. 새롭게 선택될 표준 해쉬함수는 224, 256, 384, 512 비트의 출력 크기를 지원해야 하고, 안전성 관점에서는 역상 공격, 제 2 역상 공격, 충돌쌍 공격에 대한 안전성을 요구하였다. 또한, MAC 또는 키 유도 함수 등에 사용될 경우에는 의사난수성을 제공해야 한다고 명시하였다. 본 공모 사업결과, 다양한 설계 사상을 기반으로 한 64개의 전용 해쉬함수가 제안되었으며, 여러 라운드의 평가를 거쳐 KECCAK[3]이 새로운 표준 해쉬함수 SHA-3에 선정되었다.

한편 해쉬함수에 대한 심각한 공격의 대상이 대부분 전용 해쉬함수이기 때문에, 최근에는 블록암호를 사용하는 압축/해쉬함수 운용 모드가 많은 관심을 받고 있다. 블록암호 기반 해쉬함수는 여러 가지 장점을 가진다. 안전성 측면으로 블록암호 기반 해쉬함수의 안전성은 기반 블록암호의 안전성으로 환원된다. 즉 기반 블록암호가 안전하다면, 전체 해쉬함수도 안전하다는 것을 증명할 수 있다. 효율성 측면으로는 블록암호의 키스케줄과 암호화만으로 블록암호와 해쉬함수를 모두 운영할 수 있다.

본 논문에서는 AES와 LEA 기반 이중블록길이 해쉬함수 Abreast-DM, HIROSE, MDC-2, MJH, MJH-Double에 대해 효율성을 비교 분석하였다. 그 결과, 최적화 소스코드를 이용한 AES 기반 해쉬함수보다 자체구현한 LEA 기반 해쉬함수의 속도와 메모리가 효율적이었다. 해쉬함수 관점에서는 구현결과, MJH-Double이 가장 효율적이었으며, MJH, MDC-2, HIROSE, Abreast-DM 순이었다. 구체적인 수치는 Table.3.을 참조 바라며, 속도는 cycles/byte로 측정하였다.

본 논문의 구성은 다음과 같다. 2장에서는 기 블록암호 기반 해쉬함수에 대해 소개한다. 3장에서는 AES와 LEA 기반 해쉬함수에 대한 효율성을 비교 분석하고, 4장에서는 결론 및 향후 연구방향을 제시한다.

II. 블록암호 기반 해쉬함수

대부분의 해쉬함수는 압축함수를 반복하는 구조로 설계된다. 블록암호 기반 해쉬함수는 블록암호를 기반으로 하는 압축함수를 반복하여 메시지를 처리하는 해쉬함수이다.

해쉬함수가 충분한 안전성을 가진다는 것을 보이기 위해서는 압축함수가 안전하다는 것을 보여야 한다. 안전한 압축함수를 만드는 다양한 방법이 존재하는데, 70년대 후반부터 많은 해쉬함수들이 하나 혹은 그 이상의 블록암호를 사용하는 방법으로 만들어졌고, 여전히 이런 주제와 관련한 방대한 연구가 이루어지고 있다[4].

블록암호 기반 해쉬함수는 크게 두 가지로 분류할 수 있다. 블록암호의 블록길이를 n비트라고 할 때, n비트의 해쉬값을 가지는 단일블록길이의 해쉬함수와 $2n$비트의 해쉬값을 가지는 이중블록길이의 해쉬함수로 구분할 수 있다.

대표적인 단일블록길이의 해쉬함수로는 PGV 구조가 있다[5]. Preneel, Govaerts 그리고 Vandewalle는 단일블록길이의 해쉬함수가 가질 수 있는 기본적인 64가지에 대해 분류하고 분석하였다. 단일블록길이의 경우, 충돌저항성의 안전성 수준을 128비트 혹은 그 이상으로 유지하기 위해서는 생일 역설에 의해 블록암호의 블록길이가 256비트 혹은 그 이상이 되어야하는데, 이런 블록암호는 거의 없다.

이런 문제로 인해 안전한 블록암호 기반 해쉬함수는 이중블록길이 해쉬함수에 기반한다. 대표적인 이중블록길이 해쉬함수로는 Abreast-DM, HIROSE, MDC-2, MJH, MJH-Double 등이 있다.

2.1 Abreast-DM

Abreast-DM[6]은 90년대 초반에 제안된 대표적인 블록암호 기반 해쉬함수이다. n비트 블록암호와 $2n$비트 키를 사용한다. 두 번의 블록암호 호출 각각에 다른 키가 필요하기 때문에 키 스케줄도 두 번 수행된다.

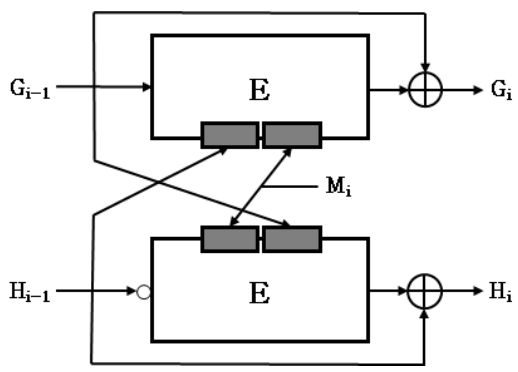


Fig. 1. Abreast-DM Compression Function

압축함수의 출력은 다음과 같이 계산된다(Fig. 1). 여기서, | 는 연접을, $\overline{X}$는 X의 보수를 의미한다.

$$G_i = G_{i-1} \oplus E_{H_{i-1}|M_i}(G_{i-1})$$
$$H_i = H_{i-1} \oplus E_{M_i|G_{i-1}}(\overline{H}_{i-1})$$

2.2 HIROSE

HIROSE[7]는 2006년 Shoichi Hirose에 의해 제안된 이중블록길이의 해쉬함수이다. HIROSE는 Abreast-DM과 동일하게 $2n$비트 키 길이의 n비트 블록암호를 2회 사용하지만, 각각의 키 스케줄에 동일한 키가 사용되기 때문에 키 스케줄 연산이 한 번 수행된다.

내부의 p 함수는 $p(p(\bullet))$이 항등 함수여야 하며 고정점을 갖지 않아야 하는 조건이 있다. 즉, p는 고정점이 없는 인볼루션 함수이다. 일반적으로 p 함수는 영이 아닌 상수 c에 대해 $p(x) = x \oplus c$를 사용한다.

압축함수의 출력은 다음과 같이 계산된다(Fig. 2).

$$G_i = E_{H_{i-1}|M_i}(G_{i-1}) \oplus G_{i-1}$$
$$H_i = E_{H_{i-1}|M_i}(p(G_{i-1})) \oplus p(G_{i-1})$$

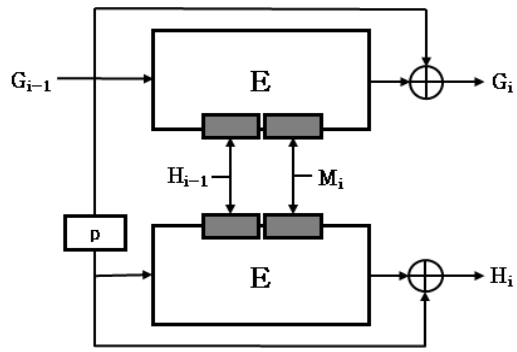


Fig. 2. HIROSE Compression Function

2.3 MDC-2

MDC-2[8]는 가장 오래된 이중블록길이 해쉬함수 중 하나로써, ANSI X9.31과 ISO/IEC 10118-2의 표준으로 규정되어 있다. 기반 블록암호로 DES를 사용하는 것으로 설계되었지만, 메시지와 키 길이가 같은 블록암호를 두 번 사용하는 압축함수로도 볼 수 있다. 암호화를 거친 결과에서 각각의 하위 $n/2$비트가 바뀌는 것이 특징이다.

압축함수의 출력은 다음과 같이 계산된다(Fig. 3).

$$V_1 = E_{G_{i-1}}(M_i) \oplus M_i$$
$$V_2 = E_{H_{i-1}}(M_i) \oplus M_i$$
$$G_i = V_1^L | V_2^R$$
$$H_i = V_2^L | V_1^R$$

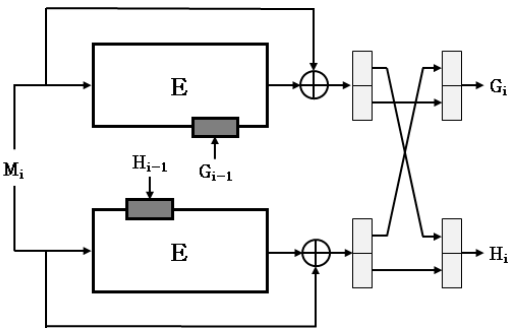


Fig. 3. MDC-2 Compression Function

단, 여기서 윗첨자로 사용된 L과 R은 각각 상위 $n/2$비트, 하위 $n/2$비트를 의미한다.

2.4 MJH

MJH[9]는 2011년에 발표된 이중블록길이 해쉬함수이다. MDC-2와 유사하게 메시지와 키 길이가

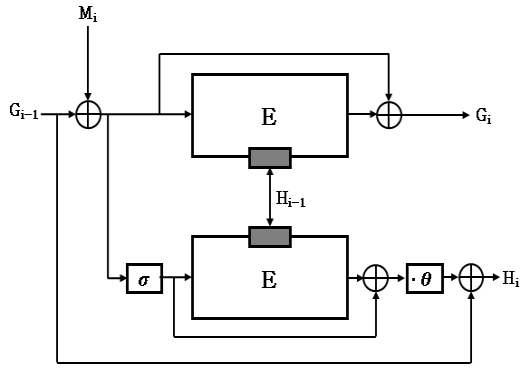


Fig. 4. MJH Compression Function

같은 블록암호를 두 번 사용한다. 하지만 각각의 키 스케줄에 동일한 키가 사용되기 때문에, 키 스케줄 연산이 한 번 수행되며, 블록암호 이외에 인볼루션 함수와 유한체 연산이 사용된다.

압축함수의 출력은 다음과 같이 계산된다(Fig. 4).

$$V_1 = G_{i-1} \oplus M_i$$
$$G_i = E_{H_{i-1}}(V_1) \oplus V_1$$
$$V_2 = E_{H_{i-1}}(\sigma(V_1)) \oplus \sigma(V_1)$$
$$H_i = (V_2 \bullet \theta) \oplus G_{i-1}$$

여기서 사용되는 σ는 고정점이 없는 인볼루션 함수여야 하며, $\bullet\theta$는 유한체 F_{2^n} 상에서 0이나 1이 아닌 상수 θ를 곱한 것을 의미한다.

2.5 MJH-Double

MJH-Double[9]은 MJH와 유사하며, 메시지의 입력을 두 배로 받는 것이 차이점이다. 즉, $2n$비트 키 길이를 가지는 블록암호를 사용한다.

압축함수의 출력은 다음과 같이 계산된다(Fig. 5).

$$V_1 = G_{i-1} \oplus M_i$$
$$G_i = E_{H_{i-1}|M_{i-1}}(V_1) \oplus V_1$$
$$V_2 = E_{H_{i-1}|M_{i-1}}(\sigma(V_1)) \oplus \sigma(V_1)$$
$$H_i = (V_2 \bullet \theta) \oplus G_{i-1}$$

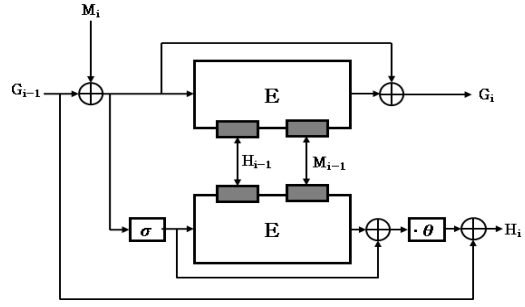


Fig. 5. MJH-Double Compression Function

III. AES/LEA 기반 이중블록길이 해쉬함수 효율성 분석

이중블록길이 해쉬함수의 효율성 분석을 위해, 속도와 메모리를 중점으로 성능을 측정한다. 성능 측정은 압축함수 단위에서 실시한다. 블록암호 기반 해쉬함수의 성능은 대부분 블록암호에 큰 영향을 받기 때문에, 먼저 블록암호에 대해 분석한다.

3.1 압축함수에 사용되는 블록암호 분석

3.1.1 AES

AES(Advanced Encryption Standard)[10]는 2001년 NIST에 의해 제정된 표준 암호 알고리즘이다. Rijndael 암호에 기반하며, 미국 정부가 AES로 채택한 이후로 전 세계적으로 가장 널리 사용되는 블록암호이다. AES는 전체적으로 SPN 구조이고, 블록길이는 128비트, 키 길이는 128/192/256 비트 3가지로 나눠진다. 라운드 수는 키 길이에 따라 각각 10/12/14 라운드를 가진다. 알고리즘 내부적으로는 SubBytes, ShiftRows, MixColumns, AddRoundKey 4개의 과정을 반복하는 구조이다. AES는 가장 널리 사용되는 점이나 안전성/효율성 등 여러 가지 요소를 고려하였을 때, 압축함수에 사용될 블록암호로 가장 적합하다.

3.1.2 LEA

LEA[12]는 2012년 국내에서 개발된 암호 알고리즘이다. 빅데이터 또는 IoT환경 등 적은 메모리와 저효율의 CPU를 가진 열악한 환경에서도 데이터를 고속으로 안전하게 암호화할 수 있는 알고리즘이다. LEA는 AES와 동일하게 블록길이는 128비트이고, 키 길이도 128/192/256비트 3가지로 나눠진다. 라

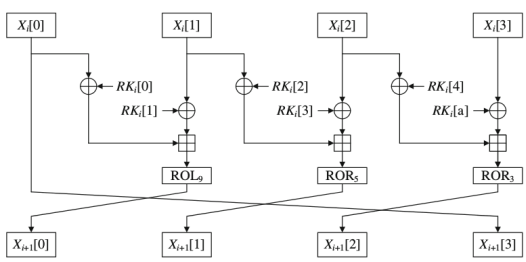


Fig. 6. LEA Encryption Round Function

운드 수는 키 길이에 따라 각각 24/28/32 라운드를 가진다. 알고리즘 내부적으로는 LEA는 비트 단위의 Addition, Rotation, XOR 3가지 연산만 사용하는 ARX구조이다. 라운드함수는 네 개의 브랜치 중 세 개의 브랜치에 대하여, 두 브랜치에 각각 라운드키를 XOR한 뒤, 다른 브랜치에 라운드키를 XOR한 값을 Addition하고, 마지막으로 Rotation을 하고 위치를 변경한다. 나머지 한 개의 브랜치는 위치만 변경한다.

3.1.3 AES/LEA 효율성 분석

AES와 LEA의 속도와 메모리요구량을 비교한다. 이중블록길이 해쉬함수에서 필요한 AES-128/256과 LEA-128/256를 분석한다.

실험환경은 일반적인 범용 PC에서 실험하였으며, 그 세부 사양으로 프로세서는 Inter(R) Core(TM) i7-4790 CPU (3.60GHz), 메모리는 16GB, 운영체제는 Windows 7 64bit이며, Visual Studio 2013의 컴파일러 ver 18.00를 사용하였다.

AES에 대한 소스코드는 오픈소스로 가장 많이 활용되는 openssl의 32비트 고속구현을 적용시킨 최적화된 AES를 사용하였다. 32비트 고속구현은 8비트 구현 속도보다 4~5배의 효율을 얻을 수 있지만[11], 메모리요구량이 5배 이상 높아지게 된다. LEA에 대한 소스코드는 자체적으로 구현한 고속구현 코드를 사용하였다.

속도 측정방법은 2014 LEA 구현경진대회에서 사용한 방법을 참고하여 총 cycles을 측정하였다. 2014 경진대회의 공개용 소스코드와 비교하였을 때, 키 스케줄은 약 3~4배(128비트 키인 경우, 342→114 cycles 향상, 256비트 키인 경우, 937→228 cycles 향상), 암호화는 약 1.4배(128비트 키인 경우, 213→151 cycles 향상, 256비트 키인 경우, 285→201 cycles 향상)의 효율을 보인다. 본 속도 향상은 LEA 자체구현에 대한 다음 특징에 기인한다. 암호화(Enc)에서는 라운드함수에서 4개의 워드의 위치가 바뀌게 되는데, 4개의 라운드를 거치면 원래 자리로 돌아오게 된다. 이 4개의 라운드를 하나로 묶어 위치를 바꾸는 불필요한 연산을 생략하였다. 위치를 바꾸지 않게 되면, 두 번째 라운드 이후로는 이전 라운드에서 전달되는 값이 원래 구조와 다른 값이 오기 때문에, 이를 고려하여 다음 라운드를 구현하였다. 키 스케줄(Key)에서는 키 스케줄 상수를 모두 사전 계산하여 소스 코드에 직접 입력하는 방식을 이용하였다. 이 방식으로 라운드 키 생성에 필요한 연산을 절반 이상 줄일 수 있다.

AES와 LEA 모두 32비트 기반으로 구현하였으며, 실험 결과는 Table. 1과 같다. 메모리는 함수가 호출될 때 필요한 변수와 함수 내에서 선언되는 변수들의 총 합을 계산하였으며, AES의 32비트 테이블은 4,096바이트의 메모리를 필요로 한다.

Table. 1의 cycles는 각각의 암호화와 키 스케줄에 사용된 cycle량이며, Spd.는 블록암호의 효율성에 해당하는 부분으로, 해쉬함수의 압축함수와 대응될 수 있는 CBC 운용 모드에서의 한 블록 처리에 대한 cycles/byte이다.

속도 면에서 LEA가 전반적으로 빠른 것을 알 수 있다. AES는 128과 256비트의 키 스케줄 속도가

Table 1. AES/LEA Speed and Memory Efficiency Comparison(Key-unrolling Method)
(Spd - cycles/byte, Mem - bytes)

		Cycles	Spd	Mem
AES-128	Enc	210	15.56	4476
	Key	117	-	
AES-256	Enc	289	20.63	4508
	Key	111	-	
LEA-128	Enc	151	12.44	80
	Key	114	-	
LEA-256	Enc	201	15.81	112
	Key	228	-	
AES-128/LEA-128 rate	Enc	1.39	1.25	55.95
	Key	1.02		
AES-256/LEA-256 rate	Enc	1.43	1.30	40.25
	Key	0.48		

Table 2. Comparison of the Compression Functions (Key/ Message length - bits, Block length - 128bit)

	Key length	# of Key Schedule	Message Length
Abreast-DM	256	2	128
HIROSE	256	1	128
MDC-2	128	2	128
MJH	128	1	128
MJH-Double	256	1	256

비슷하지만, LEA는 절반정도 차이가 난다. 하지만 키 스케줄과 암호화의 비율이 1:2인 압축함수에서는, 암호화가 빠른 LEA가 효율적이라고 볼 수 있다. 운영모드에서의 속도 또한 마찬가지이다. 메모리 면에서는 AES는 고속구현을 위한 테이블을 사용하기 때문에, LEA가 더 효율적이다.

블록암호 운용 모드(CBC, CTR 등)는 여러 개의 평문 블록을 암호화하기 위해 사용되며, 각 평문 블록은 동일한 마스터키에 의해 암호화된다. 이러한 성질로 인해, 운용 모드는 on-the-fly가 아닌 key-unrolling 방식이 적합하며, 속도측면에서 효율적이다. 왜냐하면, key-unrolling 방식으로 마스터 키에 대한 라운드 키를 선 계산 후 저장하면, 매번 평문 블록 암호화 시 호출하여 사용할 수 있기 때문이다. 반면, 블록암호 기반 해쉬함수는 압축함수에 들어가는 블록암호의 마스터 키 부분에 대부분 다른 값이 입력되기 때문에 key-unrolling 방식보다, 암호화 과정에 매 라운드 키를 생성해서 사용하는 on-the-fly 방식이 적합하고, 메모리 측면에서 더 효율적이다.

3.2 AES/LEA 기반 압축함수 성능 분석

앞서 살펴본 블록암호 AES와 LEA를 기반으로 한 해쉬함수의 성능을 측정한다. 그 대상은 Abreast-DM, HIROSE, MDC-2, MJH, MJH-Double이다.

3.2.1 압축함수별 특징

5가지의 해쉬함수는 모두 두 번의 블록암호 호출을 필요로 하는 이중블록길이 해쉬함수이다. 세부적으로 블록암호의 키 길이, 키 스케줄의 횟수, 메시지 처리량이 조금씩 다르며 Table. 2.와 같다.

본 해쉬함수 구현에서 속도에 대한 효율성을 극대화하기 위하여, 키 스케줄이 1회인 압축함수의 경우 키 스케줄 1번에 암호화가 2번 진행되는 별도의 On-the-fly 구현을 적용하였다.

3.2.2 성능 분석

앞서 언급한 블록암호 측정 환경과 동일한 환경에서 측정하였다. Table. 3.과 Table. 4.는 각각 메모리와 속도면에서 본 연구의 시험결과이다.

Table 3. Memory Comparison of a Compression Functions Efficiency

(bytes)

		AES-128	AES-256	LEA-128	LEA-256	AES/LEA rate
Hash function mode	Abreast-DM	-	4267	-	160	26.66
	HIROSE	-	4273	-	192	22.25
	MDC-2	4247	-	136	-	31.22
	MJH	4269	-	172	-	24.81
	MJH-Double	-	4321	-	240	18.00

Table 4. Speed Comparison of a Compression Functions Efficiency

(cycles/byte)

		AES-128	AES-256	LEA-128	LEA-256	AES/LEA rate
Hash function mode	Abreast-DM	-	59.25	-	63.09	0.93
	HIROSE	-	50.91	-	47.44	1.07
	MDC-2	48.47	-	41.63	-	1.16
	MJH	40.31	-	33.66	-	1.19
	MJH-Double	-	25.50	-	23.95	1.06
Block cipher Operation mode	CBC mode	15.56	-	12.44	-	1.25 (128bit)
		-	20.63	-	15.81	1.30 (256bit)

블록암호를 기준으로 분석한 결과는 다음과 같다. 일반적으로 LEA가 AES 기반 해쉬함수보다 더 효율적이었다. 속도 면에서, Abreast-DM을 제외한 모든 해쉬함수에서 LEA가 AES보다 6%~19% 정도 더 빨랐다. 메모리 면에서도 AES의 고속구현 테이블로 인해 LEA가 20~30배의 효율성을 가졌다. Abreast-DM의 경우, AES가 LEA보다 6~7% 정도 더 빠르다. 그 이유는 Abreast-DM 압축함수가 256bit 키 스케줄을 두 번 호출하기 때문이다. 앞서 보았듯이 LEA-256 키 스케줄은 AES-256 키 스케줄 보다 2배 가량 느리다.

압축함수를 기준으로 봤을 때, 128비트 키를 사용하는 MDC-2와 MJH가 256비트 키를 사용하는 Abreast-DM과 HIROSE보다 효율적인 것을 알 수 있다. 블록암호 자체의 속도가 128비트 키를 사용하는 것이 빠르며, 키를 저장하는 공간 또한 128비트 작기 때문이다. 하지만, MJH-Double의 경우 256비트 키를 사용하지만, 메시지 처리량이 두 배이기 때문에 속도 면에서 가장 효율적인 것을 알 수 있다. 다만 256비트 키를 두 개 저장해야 하기 때문에 가장 많은 메모리를 필요로 하는 결과를 보인다. 또한 동일한 키 길이를 사용하는 압축함수의 경우 키 스케줄 횟수가 적은 압축함수들이 더 빠른 결과를 보인다. Abreast-DM보다 HIROSE가, MDC-2보다 MJH가 더욱 빠른 것을 알 수 있다.

[13]에서 제시한 AES-NI 기반 해쉬함수의 효율성과 비교하였을 때, 본 논문의 AES 기반 해쉬함수에 대한 속도는 [13]과 비슷한 비율을 보인다. 하지만, LEA 기반 해쉬함수에 대한 결과를 통해, 단순히 블록암호의 효율성으로 해쉬함수의 효율성이 결정되지 않는 것을 알 수 있다. 해쉬함수의 압축함수의 속도는 블록암호의 운용모드와 비교해 볼 수 있다. 128비트에서 압축함수에 대한 AES/LEA rate와 CBC 운용 모드에 대한 rate의 차이는 약 0.1이고, 256비트에서는 0.3정도의 차이를 보인다. 이는 키 스케줄에 큰 영향을 받는 압축함수에서, LEA의 키 스케줄이 상대적으로 효율성이 떨어지고, 특히 256비트에서 차이가 더 크기 때문이다. 이는 블록암호 자체의 효율성이 해쉬함수의 효율성에 그대로 전이되지 않는다는 것을 의미한다.

결론적으로 압축함수 구조로 인한 속도 차이는 메시지 처리량, 키 길이 및 키 스케줄의 횟수에서 발생하며, 메모리요구량 차이는 앞서 살펴본 블록암호 구현방식과 압축함수 구조에 종속적으로 발생한다.

IV. 결 론

본 논문에서는 이중블록길이 해쉬함수 Abreast-DM, HIROSE, MDC-2, MJH, MJH-Double에 대한 AES 및 LEA의 효율성을 분석하였다. 구현 결과, 최적화로 고속구현 된 AES보다 LEA가 Abreast-DM을 제외한 이중블록길이 해쉬함수에서 속도와 메모리 측면에서 더 효율적이었다. 또한, 해쉬함수는 키 스케줄의 영향이 크기 때문에, 블록암호의 효율성이 그대로 해쉬함수에 전이되지 않았다.

향후 연구로는 경량환경에 적합한 해쉬함수 구현을 위해 경량으로 개발된 다양한 블록암호에 대한 해쉬함수 효율성 분석이 필요하다.

References

[1] X. Wang, H. Yu, "How to Break MD5 and Other Hash Functions," Advances in Cryptology, EUROCRYPT'05, LNCS 3494, pp. 19-35, May. 2005.

[2] X. Wang, Y. Lisa Yin, H. Yu, "Finding Collisions in the full SHA-1," Advances in Cryptology, CRYPTO'05, LNCS 3621, pp. 17-36, August. 2005.

[3] G. Bertoni, J. Daemen, M. Peeters, G. Van Assche, "Keccak sponge function family main document," Submission to NIST(Round 2), June 2009.

[4] E. Andreeva, B. Mennink, B. Preneel, "Open problems in hash function security," Designs, Codes and Cryptography, Vol. 77. Issue 2. pp. 611-631, May. 2015.

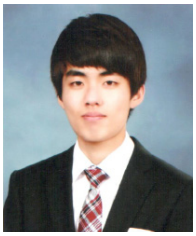
[5] B. Preneel, R. Govaerts, J. Vandewalle, "Hash functions based on block ciphers: A synthetic approach," Advances in Cryptology, CRYPTO'93, LNCS 773, pp. 368-378, August. 1994.

[6] X. Lai, J. L. Massey, "Hash function based on block ciphers," Advances in Cryptology, EUROCRYPT'92, LNCS 658, pp. 55-70, May. 1992.

[7] Shoichi Hirose, "Some plausible con-

structions of double-block-length hash functions," Proceedings of FSE'06, LNCS 4047, pp. 210-225, March. 2006.
〔8〕 B. O. Brachtl, D. Coppersmith, M. M. Hyden, S. M. Matyas Jr, Carl H. W. Meyer, Jonathan Oseas, Shaiy Pilpel, Michael Schilling, "Data authentication using modification detection codes based on a public oneway encryption function," U.S. Patent No.4908861, March. 1990.
〔9〕 Jooyoung Lee, M. Stam, "MJH: A Faster Alternative to MDC-2," Proceedings of CT-RSA'11, LNCS 6558, pp. 213-236, Feb. 2011.
〔10〕 NIST, "Announcing the ADVANCED ENCRYPTION STANDARD(AES)," FIPS PUBS 197, Nov. 2001.
〔11〕 G. Bertoni, L. Breveglieri, P. Fragneto, M. Macchetti, S. Marchesin, "Efficient Software Implementation of AES on 32-Bit Platforms," Proceedings of CHES'02, LNCS 2523, pp. 159-171, August. 2003.
〔12〕 Deukjo Hong, Jung-Keun Lee, Dong-Chan Kim, Daesung Kwon, Kwon Ho Ryu, Dong-Geon Lee, "LEA: A 128-Bit Block Cipher for Fast Encryption on Common Processors," Proceedings of WISA 2013, LNCS 8267, pp. 3-27, August. 2014.
〔13〕 Joppe W. Bos, Onur Özen, Martijn Stam, "Efficient Hashing Using the AES Instruction Set," Proceedings of CHES 2011, LNCS 6917, pp. 507-522, October. 2011.

〈저자소개〉



김 도 원 (Dowon Kim) 학생회원
2015년 2월: 국민대학교 수학과 졸업
2015년 3월~현재: 국민대학교 금융정보보안학과 석사과정
〈관심분야〉 정보보호, 암호 알고리즘, 디지털 포렌식

김 종 성 (Jongsung Kim) 종신회원
2000년 8월/2002년 8월: 고려대학교 수학 학사/이학석사
2006년 11월: K.U.Leuven, ESAT/SCD-COSIC 정보보호 공학박사
2007년 2월: 고려대학교 정보보호대학원 공학박사
2007년 3월~2009년 8월: 고려대학교 정보보호기술연구센터 연구교수
2009년 9월~2013년 2월: 경남대학교 e-비즈니스학과 조교수
2013년 3월~현재: 국민대학교 수학과 부교수
2014년 3월~현재: 국민대학교 일반대학원 금융정보보안학과 부교수
〈관심분야〉 정보보호, 암호 알고리즘, 디지털 포렌식