Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.32, NO.1, Feb. 2022 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2022.32.1.1

32-bit RISC-V 상에서의 사전 연산을 활용한 Fixslicing AES-CTR 속도 최적화 구현*

엄 시 우,[1†] 김 현 준,[1] 심 민 주,[1] 송 경 주,[1] 서 화 정[2‡]
[1,2]한성대학교 (대학원생, 교수)

Implementation of Fixslicing AES-CTR Speed Optimized Using Pre-Computed on 32-Bit RISC-V*

Si-Woo Eum,[1†] Hyun-Jun Kim,[1] Min-Joo Sim,[1] Gyeong-Ju Song,[1] Hwa-Jeong Seo[2‡]
[1,2]Hansung University (Graduate student, Professor)

요 약

Fixslicing AES는 Bitsliced AES의 선형 계층에서 많은 Cycle이 발생하는 것을 최소화하기 위해 Shiftrows 단계를 생략한 기법으로 Bitsliced 기법 대비 30% 성능 향상을 보여준다. 하지만 생략된 Shiftrows를 보완하기 위해 코드량이 증가되기 때문에 Shiftrows를 절반만 생략한 Semi-Fixsliced와 완전히 생략한 Fully-Fixsliced로 나뉜다. 본 논문에서는 사전 연산 테이블 기법을 활용한 RISC-V 상에서의 Fixslicing AES의 CTR 모드 구현을 제안한다. CTR 모드의 특징을 활용하여 2-라운드 SubBytes 연산까지의 사전 연산을 통해 암호화 과정에서 2-라운드 SubBytes까지 생략한 빠른 암호화가 가능하다. 해당 기법을 활용하여 32-bit RISC-V 상에서 Semi-Fixsliced는 하나의 블록을 암호화하는 비용은 1,345 Cycle이며 기존 대비 7%의 성능 향상, Fully-Fixsliced는 1,283 Cycle 이며 기존 대비 9%의 성능 향상을 확인하였다.

ABSTRACT

Fixslicing AES is a technique that omits the Shiftrows step to minimize the cost of the linear layer of Bitsliced AES, showing a 30% performance over the Bitsliced technique. However, the amount of code increases to compensate for the omitted shiftrows. Therefore, it is proposed to be divided into Semi-Fixsliced in which only half of shiftrows are omitted and Fully-Fixsliced in which Shiftrows are omitted completely. In this paper, we propose a CTR mode implementation of Fixslicing AES on RISC-V using the pre-computed table technique. By utilizing the characteristics of the CTR mode, it is possible to perform fast encryption by omitting up to the second round SubBytes from the encryption process through pre-computed up to the second round SubBytes operation. Using this technique, it was confirmed that Semi-Fixsliced has a performance of 1,345 cycles per block and a performance improvement of 7% compared to the previous performance result, and Fully-Fixsliced has a performance of 1,283 cycles per block and a performance of 9% compared to the previous performance result on 32-bit RISC-V.

Keywords: AES, CTR, Fixslice, Optimized implementation, RISC-V

Received(11. 01. 2021), Modified(01. 04. 2022), Accepted(01. 04. 2022)

* 본 논문은 2021년도 한국정보보호학회 충청지부 학술대회에 발표한 우수논문을 개선 및 확장한 것임

* 이 논문은 부분적으로 2021년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No.2018-0-00264, IoT 융합형 블록체인 플랫폼 보안 원천 기술 연구, 50%) 그리고 이 논문은 부분적으로 2021년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No.2021-0-00540, GPU/ASIC 기반 암호알고리즘 고속화 설계 및 구현 기술개발, 25%) 그리고 이 성과는 부분적으로 2021년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No. NRF-2020R1F1A1048478, 25%).

† 주저자, shuraatum@gmail.com

‡ 교신저자, hwajeong84@gmail.com(Corresponding author)

I. 서 론

DES의 암호화 강도가 점점 약해지면서 NIST(National Institute of Standards and Technology)에서 새로운 암호화 알고리즘을 공모하여 채택된 AES[1](Advanced Encryption Standard)는 벨기에 암호학자에 의해 개발된 Rijndael 알고리즘이다. 전 세계적으로 오랫동안 사용되어 온 암호로 연구 또한 활발하게 이루어졌다.

Bitsliced AES[2]는 함수의 계산이 논리 게이트(AND, XOR, OR 등)로 축소되어 CPU의 레지스터 폭만큼 많은 인스턴스를 병렬로 실행할 수 있도록 하는 소프트웨어 구현 기법이다. Bitsliced AES의 RISC-V에서 가장 효율적인 구현은 Schwabe와 Stoffelen[3]의 구현으로 124cpb(cycle per bit: 1-bit를 암호화하는데 필요한 cycle)의 성능을 보여준다.

최근 CHES'21에서 발표된 기법인 Fixslicing AES[4]는 기존 Bitsliced로 구현된 AES에서 선형 계층에 필요한 작업량을 이론상으로 52%까지 줄인 기법으로 RISC-V 프로세서 상에서 87cpb의 성능을 보이며 Bitsliced 구현보다 30%의 성능 향상을 보여준다.

본 논문에서는 사전 연산 테이블을 활용한 Fixslicing AES-CTR 구현을 제안한다. 본 논문의 구성은 다음과 같다. 2장에서는 CTR 운용모드, Fixslicing AES, 그리고 RISC-V 프로세서에 관해 설명한다. 3장에서는 구현 기법에 관해 설명한다. 4장에서는 성능 평가를 진행한다. 마지막으로 5장에서는 결론을 내린다.

II. 관련 연구

2.1 CTR(Counter) 운영 모드

CTR(Counter) 운영 모드는 이미 1979년 Diffie와 Hellman[5]에 의해 도입되었다. CTR 운영 모드는 블록 암호를 스트림 암호로 바꾼다. 기존 평문을 암호화 하는 것이 아닌 고정된 상수를 사용하는 Nonce 값과 변수인 Counter 값이 결합한 값을 입력으로 사용하여 출력된 값을 평문과 XOR 하는 방식으로 암호화가 진행된다. CTR 운용모드의 암호화 과정은 Fig. 1.과 같다.

CTR 모드의 장점은 소프트웨어와 하드웨어에 효율적인 구현이 가능하며, 스트림 암호의 특성을 활용하여 Count 값을 통해 특정 부분만 복호화를 할 수 있는 장점이 있다. 또한 Nonce값이 고정된 특징을 활용하여 사전 연산을 통한 최적화가 가능하다.

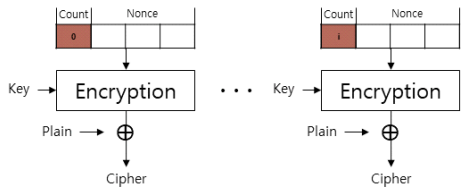

Fig. 1. Encryption process in counter mode

2.2 Fixslicing AES

AES는 128-bit의 블록 길이를 가지며, 128, 192, 256-bit 세 개의 키 길이를 지원한다. 각 키 길이에 따라 10, 12, 14 라운드를 돌며 암호화가 진행되며, 각 라운드에서는 SubBytes, Shiftrows, MixColumns, AddRoundkey의 연산을 하며 암호화가 진행된다. 전체적인 AES의 알고리즘은 Fig. 2. 와 같다.

AES의 대표적인 구현 기법인 Bitsliced AES는 RISC-V 프로세서 상에서 124cpb의 성능을 보여준다. 이때 선형 계층에 해당하는 Shiftrows는 47.5cpb가 발생하고, 이는 전체 성능에 38%에 해당한다.

Fig. 2. Round Function에서 Shiftrows 연산을 진행함에 따라 내부 데이터 상태의 변화를 보

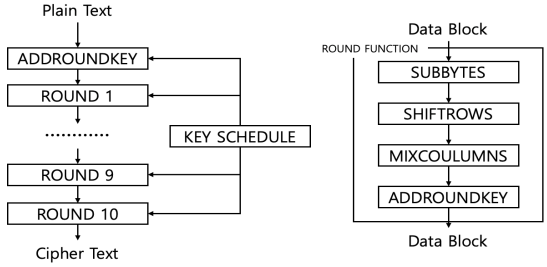

Fig. 2. Structure of AES Algorithm

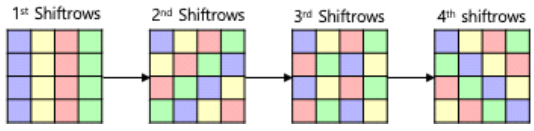

Fig. 3. Internal state change by Shiftrows operation of Round Function

면 Fig. 3.과 같이 4개의 상태가 나타나게 된다.

Fixslicing AES 기법은 Bitsliced AES기법에서의 선형 계층의 비용을 최소화하기 위해 Shiftrows 연산을 생략하고 생략된 Shiftrows 연산을 Mixcolumns에서 대체 표현하여 구현된 기법이다.

Fig. 3.처럼 Shiftrows 연산을 진행함에 따라 4개의 상태를 확인할 수 있다. 하지만 Fixslicing 기법은 Shiftrows 연산을 생략하기 때문에 Fig. 3.의 Round 0이 고정된 상태로 암호화가 진행된다. 이로 인하여 라운드가 진행됨에 따라서 Bitsliced와 Fixslicing 기법 간의 상태가 서로 달라지게 되며, 열 단위로 확산이 이루어지는 Mixcolumns 연산에서의 결과값이 달라지게 된다. 따라서 Shiftrows로 인해 변하는 4개의 상태에 맞춰 Mixcolumns 연산에서 Shiftrows 연산이 생략되기 전의 상태에 맞춰 연산이 될 수 있도록 구현이 되야한다. 결과적으로 Fixslicing 기법에서는 Shiftrows 연산을 생략하고, Mixcolumns 연산을 4개의 상태에 맞게 4개로 구현이 되어 있다. 이로 인해 Fixslicing 기법에서는 Bitsliced 기법에 비하여 선형 계층의 연산이 이론적으로 50%이상 감소한다. 하지만 4개의 Mixcolumns를 구현하기 때문에 코드량이 증가하게 된다. 따라서 Fixslicing 기법에서는 코드량을 줄이기 위하여 2번의 Shiftrows 연산을 진행하는 Semi-Fixsliced 구현과 코드량이 증가하더라고 연산량을 줄이는 Fully-Fixsliced로 나누어 구현할 수 있다.

전체적인 Fixslicing AES의 알고리즘은 Fig. 4.와 같다. Bitsliced AES의 개선된 방법이기 때문에 입력 값의 비트를 정렬하는 Packing 연산이 추가가 된다. 라운드 함수에서는 Shiftrows 연산이 생략되고 4개의 Mixcolumns 연산이 각 상태에 맞게 번갈아가며 동작한다.

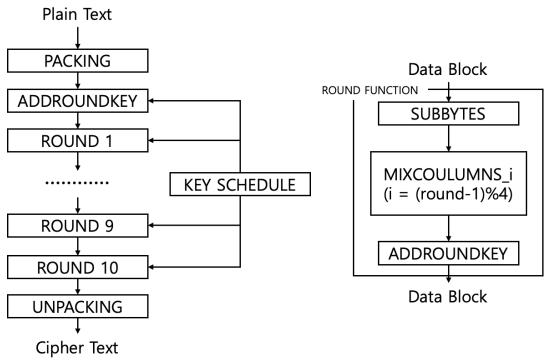

Fig. 4. Structure of Fixslicing AES

Bitsliced 기법은 병렬처리가 가능한 소프트웨어 구현 기법이다. Fixslicing 기법도 마찬가지로 병렬처리가 가능하다. Fixslicing 기법에서 32-bit 레지스터를 사용할 때, 1개의 평문 블록만 Packing을 해주게 되면 32-bit 레지스터에서 16-bit만 사용되게 된다. 따라서 32-bit를 전부 사용하기 위해 1개의 평문 블록을 더 추가하여 2개의 블록을 병렬 연산한다.

2.3 RISC-V 프로세서

RISC-V는 UC 버클리 대학교에서 2010년부터 개발중인 RISC(Reduced Instruction Set Computer) 기반의 컴퓨터 아키텍쳐이다[6]. RV32I와 RV64I의 두 가지 모델은 각각 32-bit, 64-bit 레지스터를 사용한다. 본 논문에서는 32-bit 레지스터 32개를 제공하는 RV32I를 사용한다. 레지스터 각각의 용도는 Table. 1. 과 같다[7].

zero 레지스터(x0)는 항상 0의 값을 갖는 레지스터 이다. stack pointer 레지스터(sp)는 스택의 주소를 가지고 있으며, 레지스터의 값을 스택에 넣고 뺄 때 사용되는 레지스터이다. Arguments 레지스터(a0~a7)는 함수의 인자를 포함하거나 리턴값을 가지고 있는 레지스터이다. saved 레지스터(s0~s11)는 sp(x2)와 마찬가지로 Callee saved 레지스터로 사용하기 이전의 값을 보존해야 하는 레지스터이다.

RISC-V에서 사용하는 기본적인 명령어와 본 논문에서 구현에 활용한 명령어는 Table. 2. 와 같다.

Table 1. Purpose of RISC-V Register

Register	Description	Saver
zero(x0)	Zero register	
ra(x1)	return address	
sp(x2)	stack pointer	callee
gp(x3)	global pointer	
tp(x4)	thread pointer	
a0~a7	function arguments and return value	
s0~s11	saved registers	callee
t0~t6	temporal registers	

Table 2. Instruction of RISC-V

Instruction	Description
ADD	Add
ADDI	Add Immediate
OR	Inclusive or
XOR	Exclusive or
SLLI	Shift left logical immediate
SRLI	Shift right logical immediate
LW	Load word
LBU	Load unsigned byte
SW	Store word
JAL	Jump and link
BNE	Branch if not equal

III. 제안 기법

본 논문에서 제안하는 기법은 CTR 운용 모드의 특성을 활용한 사전 연산 기법이 적용된 빠른 Fixslicing AES-CTR을 제안한다. 제안하는 기법을 적용한 Fixslicing AES-CTR의 전체적인 알고리즘은 Fig. 5와 같다.

암호화가 진행되기 전 사전 연산을 통해 사전 연산 테이블을 생성하며, Counter값에 따라서 암호화에 사용될 사전 연산된 값을 테이블에서 불러와 암호화가 진행된다. 해당 기법을 설명하기 위한 본 장의 구성은 사전 연산 기법, 사전 연산 테이블 생성, 사전 연산 테이블 적용으로 나누어 설명한다.

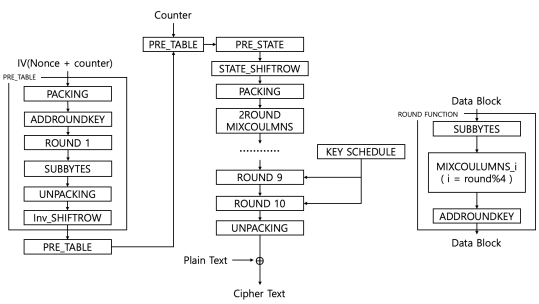

Fig. 5. Structure of Fixslicing AES-CTR using Pre-computed table

3.1 사전 연산 기법

CTR 운용 모드는 Nonce 값이 고정되는 특성이 있다. Nonce 값이 고정됨으로 인해 암호화 과정에서 일정 라운드의 특정 라운드 함수까지 연산하였을 때 고정된 값이 나오게 된다. 하지만 변수인 Counter 값으로 인해 항상 같은 값으로 고정되는 것이 아닌 Counter 값에 따라 값이 정해지게 된다.

AES의 블록 길이는 128-bit를 사용하며, 이때 Fig. 6.과 같이 고정된 상수 Nonce와 변수 Counter가 합쳐진 값인 IV(Initialization Vector)는 96-bit의 Nonce, 32-bit의 Counter로 구성된다.

Counter로 사용되는 S[0~3]의 각 State는 8-bit 크기를 가지기 때문에 0~255까지의 값을 가질 수 있으며, 각 State의 값에 따라서 사전 연산의 값이 달라지게 된다.

사전 연산이 가능한 부분을 확인하기 위해 AES 암호화 과정에서 전체 확산이 이루어지는 라운드를 알아야 한다. 전체 확산이 이루어지기 전의 State는 고정된 상수 Nonce로 인하여 Counter 값에 따라 고정된 값이 나오게 된다. AES 암호화의 확산 과정은 Fig. 7. 과 같다.

Fig. 7.은 기존 AES에서 라운드 함수가 연산됨에 따라 Counter 값에 해당하는 S[0~3]이 영향을 주는 State를 보여준다. 2라운드의 Shiftrows까지는 Counter 값들이 영향을 주는 State가 Counter값들 간에는 영향을 주지 않고 규칙적인 것을 볼 수 있다. 하지만 2라운드의 Mixcolumns 연산을 진행하게 되면 규칙이 사라지며 전체 확산이 이루어지게 된다.

따라서 전체 확산이 이루어지는 2라운드의 Mixcolunms전 단계인 Shiftrows까지 사전 연산이 가능하다. 예를 들어 Counter에 해당하는 S[0~3]의 값 중에서 S[0]의 값만 변경하고 S[1~3]의 값은 고정된 상태로 2라운드의 Shiftrows까지 연산한 결과를 확인해보면, S[0]이 영향을 주는 S[5], S[10], S[15] 값의 변화만 있을 뿐 다른 State의 값은 고정된 값이 나오게 된다.

결과적으로 Counter에 해당하는 S[0~3] 값을 0~255까지 증가시켜가며 2라운드 Shiftrow까지 사전 연산하여 암호화 과정에서 사전 연산된 부분을

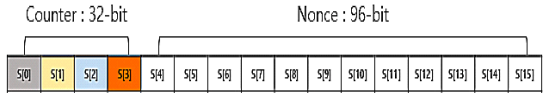

Fig. 6. IV(Counter+Nonce) State in CTR operation mode

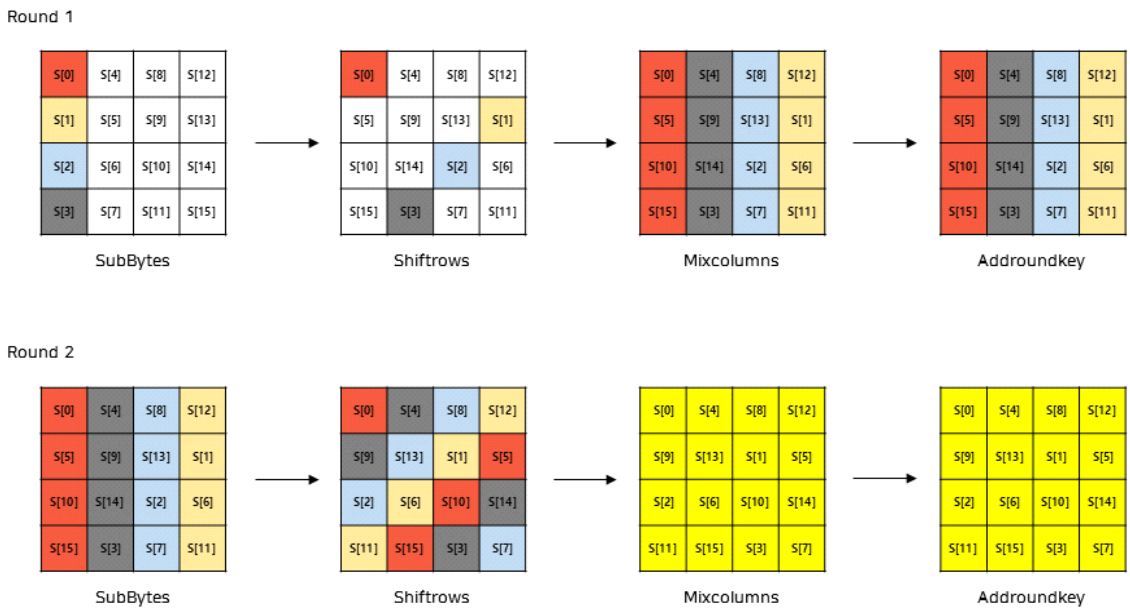

Fig. 7. The internal diffusion process of AES

생략하여 더 빠른 암호화가 가능하게 된다.

3.2 사전 연산 테이블 생성

사전 연산 테이블 생성은 Counter 값을 증가시켜 가며 전체 확산이 이루어지기 전인 2라운드의 Shiftrows까지 연산한 값을 사용한다. Fixslicing 기법을 사용하게 되면 Shfitrows 연산을 사용하지 않는다. Shiftrows 연산이 생략되었지만 결국 같은 암호화 과정이기 때문에 Fixslicing 기법도 마찬가지로 2라운드의 Mixcolumns 연산 전까지의 값을 사전 연산할 수 있다. 따라서 2라운드의 SubBytes 연산 후의 결과값을 활용하여 사전 연산 테이블을 생성한다.

각 Counter 값에 영향 받는 State는 4개이며 이는 32-bit를 의미한다. 이때 Fig. 7.에서 2라운드 Shiftrows의 결과를 보게 되면, Counter 값에 영향 받는 값들이 서로 다른 열에 분포되어 있는 것을 볼 수 있다. 이를 8-bit 단위로 테이블에 저장하게 되면 메모리 접근이 증가하기 때문에 성능 측면에서 비효율적이다. 기존의 AES에서는 2라운드의 SubBytes 까지 연산하여 저장하면 되지만 Fixslicing 기법에서는 SubBytes 연산까지 진행하게 되면 Shiftrows의 형태로 결과값이 나오기 때문에 분포되어 있는 State를 하나의 열로 모아주는 작업을 추가하여 32-bit 단위로 테이블에 저장한다. 이를 위해 Inverse Shiftrows 과정을 추가하고 테이블에 저장한다.

사전 연산 테이블에 저장할 때 Counter 값에 해당하는 S[0~3]은 서로 다른 테이블에 저장된다. 따라서 256개의 32-bit 값을 갖는 4개의 테이블이 생성되게 되므로 4KB 사전 연산 테이블이 생성된다.

사전 연산 테이블에 값을 저장할 때의 index는 현재 Counter 값을 사용한다. 예를 들어, S[0]=0x00, S[1] = 0x01, S[2] = 0x02, S[3] = 0x04 일 때, S[0]에 영향 받는 32-bit의 값은 Pre_table[0][0]에 저장되고, S[1]에 영향 받는 값은 Pre_table[1][1]에 저장된다. 즉 Pre_table[Index][Value]로 볼 수 있다.

사전 연산 테이블의 생성 과정은 Fig. 8.과 같고, 의사 코드로 보게 되면 Table 3.과 같다. Fig. 8.은 하나의 블록 상에서의 과정을 보여주고 있다. Table 3.은 Fixslicing AES에서의 의사코드로 32-bit 플랫폼에 맞춰 2개의 블록을 병렬 연산하도록 구현되어 있다. 따라서 IV_1과 IV_2 2개를 병렬 연산한다. 이때 IV_1과 IV_2의 Nonce 값은 같지만 Counter 값이 다르다. 0~255까지의 Counter 값에 따른 결과값을 저장해야 한다. 이때 짝수

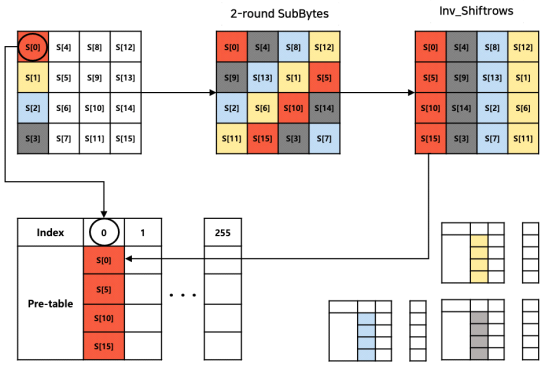

Fig. 8. Pre-table creation process using counter value

Table 3. Pseudocode of Pre-computed Table creation

Input : IV_1(Counter+Nonce), IV_2, RK(RoundKey), Pre_table[4][256]	
Output : Pre_table[4][256]	
1	**For** i = 0 to 127
2	IV_1[0] = IV_1[1] = IV_1[2] = IV_1[3] = i;
3	IV_2[0] = IV_2[1] = IV_2[2] = IV_2[3] = i+1;
4	Packing(IV_1, IV_2, State);
5	Addrounkey(State, RK);
6	SubBytes(State);
7	MixColumns_0(State);
8	Addrounkey(State, RK);
9	SubBytes(State);
10	Unpacking(State, Out_1, Out_2);
11	Inv_Shiftrow(Out_1, Out_2);
12	Store_Pretable(Out_1, Out_2, Pre_table);
13	**End For**
14	**return** Pre_table;

Counter 값은 IV_1에 홀수 Counter 값은 IV_2에 저장하여 2개의 Counter 값에 따른 결과값을 얻을 수 있다. 또한 Counter값에 해당하는 S[0~3]의 값이 2 Round의 SubBytes 까지 연산을 진행했을 때 영향을 주는 State는 서로 독립적인 걸 볼 수 있다. 따라서 Counter 값을 1씩 증가시키지 않고 각각의 Counter state(S[0~3])의 값을 모두 1씩 증가시켜 진행한다(즉, S[0], S[1], S[2], S[3] 모두 1씩 증가시킨다.). 결과적으로 128번 반복하여 좀 더 빠르게 테이블 생성을 할 수 있다.

3.3 사전 연산 테이블 적용

사전 연산 테이블에서 값을 가져올 때, 2개의 블록을 병렬 연산하기 위해 2개의 블록을 가져와야 한다. 저장할 때와 마찬가지로 현재 Counter에 해당하는 값을 Index로 사용하여 값을 불러온다. 이때 IV_1과 IV_2의 차이는 Counter 값이 1 증가한 차이만 존재한다. 즉 Count가 증가했을 때 값이 증가하는 State가 S[0]이라면, IV_1과 IV_2의 S[1~3]은 같은 값을 가지고 있고 S[0]의 값만 1 증가한 차이만 있다.

메모리 접근을 줄이기 위해 IV_1의 State를 불

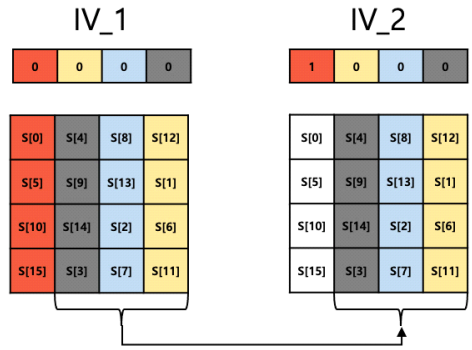

Fig. 9. Generation of IV_1 and IV_2

Table 4. Pseudocode of Fixslicing AES-CTR

Input : PT_1(Plain Text), PT_2 Counter, RK(Roundkey), Pre_table	
Output : Cipher_1, Cipher_2	
1	PreState_init(IV_1, IV_2, Pre_table, Counter);
2	Packing(IV_1, IV_2, State);
3	Mixcolumns_1(State);
4	AddRoundkey(State, RK);
5	**For** i = 3 to 10
6	SubBytes(State);
7	Mixcolumns_n(State); // n=(i%4)-1
8	AddRoundkey(State, RK);
9	**End For**
10	Double_Shiftrows(State);
11	Unpacking(State, Cipher_1, Cipher_2);
12	Cipher_1 = PT_1 XOR Cipher_1;
13	Cipher_2 = PT_2 XOR Cipher_2;
14	**retrun** Cipher_1, Cipher_2;

러오고, IV_2는 IV_1의 1개의 열을 제외한 3개의 열을 복사하는 것으로 생성할 수 있다. 이 과정은 Fig. 9.와 같다.

사전 연산된 값은 2라운드의 SubBytes까지 연산이 되어 있다. 하지만 테이블에 저장하기 위해 Inv_Shiftrows를 했기 때문에 값을 불러와서 바로 암호화를 진행할 수 없다. 따라서 값을 불러와 Shiftrows를 한번 진행하고 암호화가 진행된다.

사전 연산 테이블을 활용한 Fixslicing AES-CTR의 전체적인 의사코드는 Table 4.와 같다.

Table 4의 1번 줄은 테이블에 저장된 사전 연산 값을 불러와 저장하는 값이다. 이때 현재 Count 값을 기준으로 테이블에서 값을 불러온다. 불러 온 값을 Packing으로 내부 정렬을 진행한 후에 사전 연산되지 못한 연산들을 진행한다.

IV. 성능 평가

RISC-V는 SiFive사의 HiFive1 Rev B 플랫폼이다. 4MB의 Quad SPI 플레시 메모리, E31 코어가 탑재되어 있어 320MHz로 연산이 수행된다. 구현은 SiFive사에서 제공하는 Freedom Studio 프레임워크를 사용하였다.

성능 비교는 CTR모드로 동작한 128비트 2개의 블록을 암호화 할 때의 평균 Cycle을 측정하여 블록 수로 나눈 Cycle값으로 비교한다. 기존 연구의 성능과 비교하기 위하여 사전 연산 테이블을 생성하는 과정의 Cycle은 측정하지 않으며, 암화화 과정의 Cycle만 측정하여 비교를 진행한다. Cycle 측정은 암호화 함수를 10,000번 반복 실행시켜 측정된 Cycle의 평균 값을 사용하였다. Fig. 10은 2018년

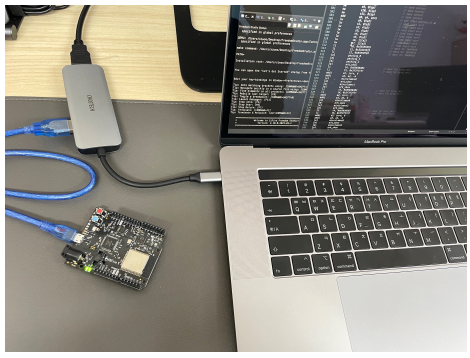

Fig. 10. RISC-V Performance Evaluation

형 MacBook Pro에 HiFive1 Rev B 보드를 연결하여 성능을 측정하고 있다.

Table 5.는 제안하는 기법을 활용한 Fixslicing AES를 CTR 모드로 동작시켰을 때의 CPB(Cycle per Block)이다. 본 장에서 사용하는 단위 CPB(Byte)는 이전에 사용한 cpb(Bit)와 단위가 다르다. 이는 기존 연구[4]에서 성능 측정 단위를 CPB를 사용하였기 때문에 본 논문에서도 CPB 단위로 성능 결과를 제시한다. 기존 연구 결과[4]에서 보여준 Semi-Fixsliced와 Fully-Fixsliced간 성능 차이는 제안한 기법의 결과에서도 비슷한 차이를 확인할 수 있다. 결과적으로 제안하는 기법을 활용하여 기존 성능 대비 Semi-Fixsliced AES-CTR은 7%, Fully-Fixsliced AES-CTR은 9%의 성능 향상을 보여준다.

Table 5. Performance result

	[4]	**This work**
Semi-Fixsliced	1447	**1345**
Fully-Fixsliced	1398	**1283**

V. 결 론

본 논문에서는 사전 연산 테이블을 활용하여 Fixslicing AES의 CTR 운용 모드 구현을 제안한다. 사전 연산은 2라운드의 SubBytes까지 진행하여 암호화 과정에서 해당 연산까지 생략한 암호화가 가능하다. 결과적으로 Pre-table은 4KB의 메모리를 사용하며, Inv_Shiftrows와 Shiftrows의 연산이 추가되지만 Mixcolumns 1번, Addroundkey 2번, SubBytes 2번의 연산을 생략할 수 있다. 제안하는 기법을 통해 RISC-V상에서 Semi-Fixsliced AES-CTR은 7%, Fully-Fixsliced AES-CTR은 9%의 성능 향상을 확인 할 수 있었다. 향후 과제로 AES와 비슷한 알고리즘을 활용하는 다른 암호에 해당 기법을 적용한 구현을 제안한다.

References

[1] J.Daemen and V.Rijmen, “Reijndael: The Advanced Encryption Standard,” *Dr. Dobb's Journal*, Mar. 2002.

[2] C.Rebeiro D.Selvakumar and A.S.L. Devi, "Bitslice Implementation of AES," *CANS 2006: Cryptology and Network Security*, pp. 203-212, Dec. 2006.

[3] P.Schwabe and K.Stoffelen, "All the AES You Need on Cortex-M3 and M4," *In Selected Areas in Cryptography*, pp. 180 - 194, Aug. 2016.

[4] A.Adomnicai and T.Peyrin, "Fixslicing AES-like ciphers: New bitsliced AES speed records on ARM-Cortex M and RISC-V," IACR-ePrint 2020-1123, Oct. 2020.

[5] W.Diffie and M.Hellman. "Privacy and Authentication: An Introduction to Cryptography," *Proceedings of the IEEE*, pp. 397 - 427, Dec. 1979.

[6] A.Waterman, Y.Lee, D.Patterson, and K. Asanovi´c, "The risc-vinstruction set manual, volume i: Baseuser-level isa," UCB/EECS-2011-62 116, UCBerkeley Tech., May. 2011.

[7] A.Waterman and K.Asanovi´c, "The RISC-V Instruction Set Manual Volume I: User-Level ISA Document Version 2.2," UCBerkeley Tech., May. 2017.

〈저자소개〉

엄 시 우 (Si-Woo Eum) 학생회원
2021년 3월~현재: 한성대학교 IT융합공학부 졸업
2021년 3월~현재: 한성대학교 IT융합공학부 석사과정
〈관심분야〉 암호구현, 정보보안

김 현 준 (Hyun-Jun Kim) 학생회원
2019년 3월: 한성대학교 IT응용시스템공학부 졸업
2021년 3월: 한성대학교 IT융합공학부 석사
2021년 3월~현재: 한성대학교 정보컬퓨터공학과 박사과정
〈관심분야〉 부채널분석, 블록체인

심 민 주 (Min-joo Sim) 학생회원
2021년 3월~현재: 한성대학교 IT융합공학부 졸업
2021년 3월~현재: 한성대학교 IT융합공학부 석사과정
〈관심분야〉 정보보호, 부채널분석

송 경 주 (Gyeong-ju Song) 학생회원
2021년 3월~현재: 한성대학교 IT융합공학부 졸업
2021년 3월~현재: 한성대학교 IT융합공학부 석사과정
〈관심분야〉 정보보호, 암호, 양자컴퓨터

서 화 정 (Hwa-Jeong Seo) 종신회원
2010년 3월: 부산대학교 컴퓨터공학과 졸업
2012년 3월: 부산대학교 컴퓨터공학과 석사
2016년 3월: 부산대학교 컴퓨터공학과 박사
2016년~2017년: 싱가포르 과학기술청 연구원
2019년~현재: 한성대학교 IT융합공학부 조교수
〈관심분야〉 정보보호, 암호화 구현, IoT