Fixslicing PIPO 구현

김현준*, 심민주*, 서화정**

*한성대학교 (대학원생)

**한성대학교 (교수)

Implementation of Fixslicing in PIPO

Hyun-jun Kim*, Min-joo Sim* Hwa-seong Seo**

*Information and Computer Engineering(Graduate student)

**Department of IT Convergence Engineering(professor)

요 약

ICISC'20에서 발표된 PIPO 경량 블록 암호 제품군은 8bit 임베디드 환경에서 우수한 성능 수치를 제공한다. 또한 비트슬라이싱 구현이 가능하도록 설계되어있다. 최근에는 블록 암호를 매우 효율적인 Constant-Time 구현이 가능한 새로운 비트슬라이싱 기법인 픽스슬라이싱이 등장하였다. 본 논문에서 비트슬라이싱 구현이 가능하도록 설계된 PIPO를 픽스슬라이싱 기법을 적용하여 속도 최적화한다. 제안기법을 적용하여 PIPO-64/128을 8bit 마이크로 컨트롤러 ATmega128에서 구현하여 1,501사이클로 암호화될 수 있음을 보인다. 이는 동일한 환경에서 가장 빠른 구현보다 13% 향상된 속도이다.

I. 서론

ICISC'20에서 발표된 PIPO[] 경량 블록 암호 제품군은 8bit 임베디드 환경에서 매우 우수한 성능을 보인다. 특징적으로 비트슬라이싱 구현이 가능하도록 설계되어 효율적인 구현이 가능하다. 최근 새로운 비트슬라이싱 기법 픽스슬라이싱[2][3]이 CHES2020에서 발표되었다. Gift와 AES 암호에 적용되어 기존 Constant-Time 구현과 비교하여 높은 속도 향상을 보였다. 본 논문에서는 PIPO에 대하여 높은 속도 향상을 위해 픽스슬라이싱 기법을 고려하였다. PIPO에 픽스슬라이싱을 적용하였을 때 이점을 연구하여 PIPO에 대한 2라운드의 새로운 표현을 제안한다.

결과적으로 PIPO-64/128을 8bit 마이크로 컨트롤러 ATmega128에서 1,501사이클로 암호화될 수 있음을 보여 동일한 환경에서 가장 빠른 구현[1] 보다 13%의 속도 향상을 달성하였다.

II. PIPO block cipher

PIPO는 ICISC'20에서 발표된 경량 블록 암호 알고리즘이다. 새로운 S-box를 사용하는 S-Layer와 바이트 단위의 비트 회전을 사용하는 비트 순열로 구성된다. 비선형 함수의 차동 및 선형 분기 수가 높게 설계되어 경량 환경에서 효율적이고 확산이 적은 비트 순열을 안전하게 사용 가능 하다. 11개의 비선형 비트 연산과 23개의 선형 비트 연산만 사용하는 비트 슬라이싱 구현도 제공된다. 바이트 단위 비트 슬라이싱 작업과 적은 수의 비선형 비트 작업으로 인해 8비트 AVR 마이크로컨트롤러의 부채널 보호 및 비보호 환경에서 기존 경량 블록 암호보다 성능이 뛰어나다.

III.픽스슬라이싱(Fixslicing)

픽스슬라이싱은 새로운 비트슬라이싱 기법으로 Adomnicai et. al[2][3]이 처음으로 제안하였

다. 하드웨어 지향 블록 암호 GIFT를 몇 번의 회전만을 사용하여 새롭게 표현하고 비트슬라이싱을 적용하여 소프트웨어 매우 효율적으로 구현하였다. 이후 Adomnicai et. al은 AES에 픽스슬라이싱 기법을 적용하고 AES와 유사한 암호에 픽스슬라이싱이 적용될 수 있음을 보였다. []에서 AES 유사 암호의 성능향상을 배경으로 PIPO의 속도 향상 구현을 위해 픽스슬라이싱 기법을 적용 하였다.

IV. 제안기법

본 논문에는 PIPO 높은 속도 향상 구현을 위해 픽스이싱을 적용한다. 이를 위해 PIPO의 특징을 연구하였다. 그림 1의 11에서 15까지와 같이 PIPO의 SBOX는 마지막은 비트 교환이 이루어진다. 마지막 단계의 비트 교환으로 (0, 3, 4, 5, 6, 2, 1, 7)의 순서에서 (7, 6, 5, 4, 3, 2, 1, 0)로 정렬된다.

```
PIPO_SBOX(X0, X1, X2, X3, X4, X5, X6, X7)
1.   X5 ^= (X7 & X6); X4 ^= (X3 & X5);
2.   X7 ^= X4; X6 ^= X3;
3.   X3 ^= (X4 | X5); X5 ^= X7;
4.   X4 ^= (X5 & X6); X2 ^= X1 & X0;
5.   X0 ^= X2 | X1; X1 ^= X2 | X0;
6.   X2 = ~X2; X7 ^= X1;
7.   X3 ^= X2; X4 ^= X0;
8.   X6 ^= (X7 & X5); T0 = X7^X6;
9.   X6 ^= (X4 | X3); T1 = X3^X5
10.  X5 ^= (X6 | X4); X2 ^= T0;
11.  X[2] ^= T[0];  T[0] = X[1] ^ T[2];
12.  X[1] = X[0]^T[1];  X[0] = X[7];
13.  X[7] = T[0];  T[1] = X[3];
14.  X[3] = X[6];  X[6] = T[1];
15.  T[2] = X[4];  X[4] = X[5];  X[5] = T[2];
```

(그림 1) Classic S-Box 구현

우리는 S-box에서 수행되는 비트 교환 처리를 제거할 수 있음을 발견하였다. 비트슬라이싱에서 비트 교환은 레지스터 간의 값 교환으로 처리 가능하다. PIPO는 비트슬라이스 구현으로 설계되어있으므로 동일하게 S-box의 비트 교환은 레지스터 간의 값 교환으로 처리된다. 해당연산을 제거, 변경한 구현은 그림 2의 11~12와 같다.

```
PIPO_SBOX(X0, X1, X2, X3, X4, X5, X6, X7)
1.   X5 ^= (X7 & X6); X4 ^= (X3 & X5);
2.   X7 ^= X4; X6 ^= X3;
3.   X3 ^= (X4 | X5); X5 ^= X7;
4.   X4 ^= (X5 & X6); X2 ^= X1 & X0;
5.   X0 ^= X2 | X1; X1 ^= X2 | X0;
6.   X2 = ~X2; X7 ^= X1;
7.   X3 ^= X2; X4 ^= X0;
8.   X6 ^= (X7 & X5); T0 = X7^X6;
9.   X6 ^= (X4 | X3); T1 = X3^X5
10.  X5 ^= (X6 | X4); X2 ^= T0;
11.  T2 = X7; X1 ^= X4 ^ (T1 & T0);
12.  X0 = X0^T1;
```

(그림 2) 제안 기법의 S-Box 구현

이를 바탕으로 PIPO를 새로운 픽스슬라이싱 표현으로 나타낸다. 먼저 변경 조건을 고려하여 함수의 변경한다. 변경된 S-Layer 이후의 R-layer의 입력은 블록 (7, 6, 5, 4, 3, 2, 1, 0)를 (0, 3, 4, 5, 6, 2, 1, 7)로 바꾸어 기존의 각 블록에 대한 8bit 왼쪽 회전 연산은 (0, 7, 4, 3, 6, 5, 1, 2)는 (2, 7, 4, 1, 5, 6, 3, 0)으로 변경한다. 다음 라운드의 Addroundkey 또한 변경을 고려하여 라운드 키의 블록 (0, 3, 4, 5, 6, 2, 1, 7)순으로 xor한다. 다음 라운드의 S-Box 연산은 블록의 입력 순서를 바꾸어 (0, 3, 4, 5, 6, 2, 1, 7)의 순서로 입력한다. 예를 들어 그림 2에서 상태 Y(0~7)를 함수에 입력 할 때 입력 X0에는 Y7, 입력 X1에는 Y1이 입력된다. 제거된 비트 교환으로 인해서 (0, 3, 4, 5, 6, 2, 1, 7)의 순서에서 (7, 6, 5, 4, 3, 2, 1, 0)로 다시 재정렬 되므원래의 상태로 돌아가게 된다. 결과적으로 추가적인 연산 명령어 없이 S-Box의 비트 교환을 제거가능하다. 다음 과정은 변경된 표현의 2라운드의 과정을 묶어 반복하여 연산한다.

PIPO는 다른 블록암호에 대한 픽스슬라이싱과 다르게 Permuation Layer 변형 대신 Substitution Layer가 변경된다. 또한 다른 대

부분의 블록 암호는 픽스슬라이싱 적용을 위해 패킹과 언패킹의 과정이 필요하다. 이는 추가적인 연산을 요구한다. 그러나 PIPO는 효율적인 비트슬라이싱 구현이 가능하도록 설계되어 이러한 패킹과 언패킹의 과정이 필요 없다. 따라서 다른 암호보다 Fixslicing 적용이 효율적이다.

V. 제안기법

이 장에서는 제안기법을 구현하여 비교한다. 먼저 제안 기법을 8bit avr 어셈블리 코드로 작성하였다. Clock cycle을 기준으로 속도를 비교하였으며 Microchip Studio의 ATmega128보드의 시뮬레이터를 사용하여 측정하였다. 기존 연구와 비교하였을 때 결과는 표 1과 같다. 제안기법은 1,501cycle을 달성하였고 이는 기존 구현의 1,574cycle 보다 13% 향상된 결과이다.

다음으로 제안기법의 다른 임베디드 환경에서의 결과를 비교하기 위해 C코드로 구현하여 32bit cortex-m3 보드에서 성능을 비교하였다. 결과는 표 1과 같다. 제안 기법은 2,216cycle을 달성하였고 이는 기존 구현의 3,404 cycle보다 53% 향상된 성능이다. 제안기법이 32bit환경에서도 높은 성능향상을 보여주는 것을 확인 할 수 있었다. 제안 기법을 어셈블리 코드로 작성할 경우 더 높은 속도의 성능을 나타낼 것으로 보이며 추후 연구를 진행할 예정이다.

	Language	Classic	Fixslicing
ATmega128	Assembly	1,574	1,501
Cortex-m3	C	3,404	2,216

표 1 제안기법의 AVR, ARM 임베디드 환경에서 성능비교

VI. 결론

본 논문은 PIPO 경량 블록 암호의 속도 향상을 위해 픽스슬라이싱 기법을 적용하였다. PIPO에 픽스슬라이싱을 적용하였을 때 이점을 연구하여 PIPO에 대한 2라운드의 새로운 표현을 제안한다. 결과적으로 PIPO-64/128을 8bit 마이크로컨트롤러 ATmega128에서 1501사이클로 암호화될 수 있음을 보여 동일한 환경에서 가장 빠른 구현 보다 13%의 속도 향상을 달성하였다. 해당 기법의 C 구현을 32bit cortex-m3 보드에 적용하였을 때 53%의 속도향상을 보였다. 추후 연구로는 어셈블리를 사용하여 cortex-m3 마이크로컨트롤러에 적용한 최적화와 다른 여러 환경에 최적 구현하여 구현 환경에 따른 성능을 연구하고자한다.

VII.Acknowledgement

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (No.2018-0-00264, Research on Blockchain Security Technology for IoT Services, 100%).

[참고문헌]

[1] Kim, Hangi, et al. "PIPO: A lightweight block cipher with efficient higher-order masking software implementations." International Conference on Information Security and Cryptology. Springer, Cham, 2020.

[2] Adomnicai, Alexandre, Zakaria Najm, and Thomas Peyrin. "Fixslicing: A new gift representation." Cryptology ePrint Archive (2020).

[3] Adomnicai, Alexandre, and Thomas Peyrin. "Fixslicing AES-like ciphers: New bitsliced AES speed records on ARM-Cortex M and RISC-V." Cryptology ePrint Archive (2020).