KIPS Trans. Comp. and Comm. Sys.
Vol.8, No.9 pp.225~230 pISSN: 2287-5891

https://doi.org/10.3745/KTCCS.2019.8.9.225

Implementation of Optimized 1st-Order Masking AES Algorithm Against Side-Channel-Analysis

Kim Kyung Ho† · Seo Hwa Jeong††

ABSTRACT

Recently, with the development of Internet technology, various encryption algorithms have been adopted to protect the sensing data measured by hardware devices. The Advanced Encryption Standard (AES), the most widely used encryption algorithm in the world, is also used in many devices with strong security. However, it has been found that the AES algorithm is vulnerable to side channel analysis attacks such as Differential Power Analysis (DPA) and Correlation Power Analysis (CPA). In this paper, we present a software optimization implementation technique of the AES algorithm applying the most widely known masking technique among side channel analysis attack methods.

Keywords : AES, Side-Channel-Attack, Masking, Optimization

부채널 분석 대응을 위한 1차 마스킹 AES 알고리즘 최적화 구현

김 경 호† · 서 화 정††

요 약

최근 사물인터넷 기술의 발전과 함께 하드웨어 디바이스에서 측정하는 센싱 데이터를 보호하기 위해 다양한 방식의 암호화 알고리즘을 채택하고 있다. 그 중 전 세계에서 가장 많이 사용하는 암호화 알고리즘인 AES(Advanced Encryption Standard) 또한 강력한 안전성을 바탕으로 많은 디바이스에서 사용되고 있다. 하지만 AES 알고리즘은 DPA(Differential Power Analysis), CPA(Correlation Power Analysis) 같은 부채널 분석 공격에 취약하다는 점이 발견되었다. 본 논문에서는 부채널 분석 공격 대응방법 중 가장 널리 알려진 마스킹 기법을 적용한 AES 알고리즘의 소프트웨어 최적화 구현 기법을 제시한다.

키워드 : AES, 부채널, 마스킹, 최적화

1. 서 론

사물인터넷 기술의 발전으로 다수의 디바이스들이 하나의 네트워크 안에서 연결되어 센싱 데이터를 서로 주고받을 수 있게 되었다. 디바이스에서 측정되는 데이터는 상황에 따라 보안이 필요한 데이터일 수도 있기 때문에 네트워크 통신에서는 데이터를 안전하면서도 빠르게 송수신할 수 있어야 한다. 이를 위해서 현재 대부분의 사물인터넷 기기에 AES 암호화 알고리즘을 포함한 다양한 종류의 경량 암호화 알고리즘을 적용하여 데이터를 빠르고 안전하게 전송하고 있다[1, 2].

그러나 암호 알고리즘 자체의 취약점이 아닌 하드웨어가 데이터를 암호화 하는 과정에서의 전력 소모량 분석, 시차 분석[3], 오류 주입[4], 전자기파 분석[5] 등 암호화 과정에서 나오는 부가적인 정보들을 이용하여 암호화된 정보를 탈취할 수 있는데 이러한 공격을 부채널 분석 공격이라고 한다.

현재 가장 많은 연구가 진행되는 분야는 전력 분석을 이용한 부채널 공격 및 그에 대한 대응 방안으로 세계적으로 많이 사용되는 암호인 AES[6], RSA(Rivest, Shamir, and Adelman)[7] 뿐만아니라 LEA[8, 9], ARIA[10] 등 다수의 국내 암호에 대한 공격 및 대응 방안 연구가 진행되고 있다[11-13].

※ 본 연구는 부분적으로 과학기술정보통신부 및 정보통신기획평가원의 대학ICT 연구센터지원사업의 연구결과로 수행되었음(IITP-2019-2014-1-00743*). 그리고 이 성과는 부분적으로 2019년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. NRF-2017R1C1B5075742).
※ 이 논문은 2019년도 한국정보처리학회 춘계학술발표대회에서 '부채널 분석 대응을 위한 1차 마스킹 AES 알고리즘 최적화 구현'의 제목으로 발표된 논문을 확장한 것임.
† 준 회 원 : 한성대학교 IT융합공학과 석사과정
†† 종신회원 : 한성대학교 IT융합공학부 조교수
Manuscript Received : July 5, 2019
First Revision : August 5, 2019
Accepted : August 7, 2019
* Corresponding Author : Seo Hwa Jeong(hwajeong84@gmail.com)

이러한 부채널 분석에 대응하기 위해 마스킹[14, 15], 랜덤 지연시간 삽입[16] 등의 다양한 대응 방안을 연구하고 있고 그 중 마스킹을 이용한 대응 기법에 대한 연구가 가장 많이 진행되고 있다. 마스킹 대응 기법은 암호화 알고리즘의 특정 연산 마다 암호화할 데이터와 난수 값을 XOR(eXclusive OR) 연산을 이용하여 데이터를 변환해서 공격자가 전력 분석을 할 수 없도록 하는 방법이다. 암호화 과정에서 불필요한 연산이 들어가기 때문에 마스킹 된 AES 알고리즘의 암호화 속도는 마스킹 하지 않은 AES 알고리즘에 비해 느리다. 하지만 이는 전력 소모량 분석을 이용한 부채널 공격에 대한 안전성을 확보할 수 있다.

본 논문에서는 전력 분석을 이용한 부채널 분석 공격에 대응하기 위한 1차 마스킹 기법을 AES 알고리즘에 적용하고 다른 경량 암호에 비해 암호화 시간이 긴 AES 알고리즘의 단점을 보완하기 위한 소프트웨어 최적화 구현 기법을 제시한다.

본 논문의 구성은 다음과 같다. 2장에서는 대표적인 전력 분석을 이용한 부채널 공격 방법 소개와 대응 방안인 1차 마스킹 기법을 적용한 AES 알고리즘에 대해 설명하고 3장에서는 제안하는 최적화 구현 기법을 제시한다. 4장에서는 키 길이별로 최적화된 AES 알고리즘의 성능 분석 결과를 확인해 본다. 마지막으로 5장에서는 본 논문의 결론을 내린다.

2. 전력 분석을 이용한 공격과 1차 마스킹 기법

본 장에서는 전력 소모량 분석을 이용한 부채널 공격으로 가장 잘 알려진 DPA[17]와 CPA[18] 공격에 대한 설명과 그에 대한 대응 방안인 1차 마스킹 기법을 적용한 AES 암호화 알고리즘에 대해서 서술한다.

2.1 전력 분석을 이용한 부채널 공격

부채널 분석은 암호화 알고리즘을 사용하는 디바이스의 전력 소모량의 변화를 관찰하여 통계적인 분석을 통해 키 값을 알아낸다. 즉 디바이스가 키 값과 관련이 있는 데이터의 0과 1값을 연산할 때 두 값의 전력 사용량의 차이를 이용한다.

가장 많이 사용되는 전력 분석을 이용한 부채널 공격 방식으로는 DPA와 CPA 공격 기법이 있다. DPA와 CPA는 암호화 알고리즘이 적용된 디바이스에 동일한 키 값을 이용해서 다른 평문을 연속적으로 입력하여 암호문을 얻고 전력 분석 모듈을 이용하여 파형을 수집한다. 수집된 정보들을 바탕으로 정해진 분류 함수를 이용하여 파형 통계 처리로 키 값을 분석하는 방식이다.

DPA는 평문과 분석하는 부분의 출력 값 등의 정보를 이용하여 결과 값을 0과 1 그룹으로 파형을 나누고 그룹 간 평균값을 구하여 두 그룹의 평균값의 차이로 키 값을 분석하는 방식이고 CPA는 알고리즘의 특정 연산 부분을 지정하여 분석한 다음 분석된 데이터를 이용하여 중간값 추정치를 예측한다. 추정치를 이용하여 예상되는 전력 소모 모델을 설계하고 실제 전력소모 모델과 비교하여 상관 계수가 높은 값을 키 값으로 분석한다.

2.2 1차 마스킹 AES

기본적인 AES 알고리즘은 키의 길이에 따라 128bit, 192bit, 256 bit로 나누어지며 128bit은 10번의 라운드, 192bit은 12번의 라운드, 그리고 256bit는 14번의 라운드에 걸쳐서 사용할 라운드 키를 계산하고 AddRoundKey, SubBytes, ShiftRows, MixColumns 연산을 반복하여 암호문을 얻게 되는 전 세계적으로 가장 많이 사용하는 대칭키 암호화 알고리즘이다. 본 논문에서는 키의 길이에 따라 연산의 큰 차이가 없기 때문에 AES-128을 기준으로 서술한다.

AES 알고리즘의 암호화 과정에서 공격자는 전력분석을 통해 CPA 공격을 이용하여 특정 연산의 데이터 값을 분석하고 전력 소모 모델을 만들어서 상관 계수가 높은 키 값을 찾아낸다. 따라서 CPA 공격에 대응하기 위하여 임의 값으로 이루어진 마스킹 값을 암호화 과정마다 반복적으로 적용해 전력분석을 방해하여 공격자의 CPA 공격을 방어할 수 있다[19].

마스킹 AES 알고리즘을 보기 쉽게 도식화하면 Fig. 1과 같다. 우선 마스킹 적용을 위해서 AES 알고리즘이 시작될 때 6개의 난수 값을 이용하여 M0, M1, M2, M3, M4, M5 변수를 만든다. 그리고 MixColumns 연산의 인자에 M2, M3, M4, M5를 입력하여 나오는 출력값을 M6, M7, M8, M9 값으로 사용한다. 마스킹 값에 MixColumns 연산을 하는 이유는 라운드 내에서 MixColumns 연산이 수행될 때 암호문에 마스킹된 M2, M3, M4, M5 값을 M6, M7, M8, M9로 바꿔주기 위함이다. 그 다음 SubBytes 연산에서 마스킹 연산을 추가하기 위해 Sbox 테이블을 Algorithm 1과 같은 방법으로 마스킹 한다. 이제 생성된 10개의 임의의 마스크 값(M0~M9)과 마스킹 된 Sbox 테이블을 이용해 라운드 별로 이루어지는 특정 연산에 마스킹 연산을 추가한다.

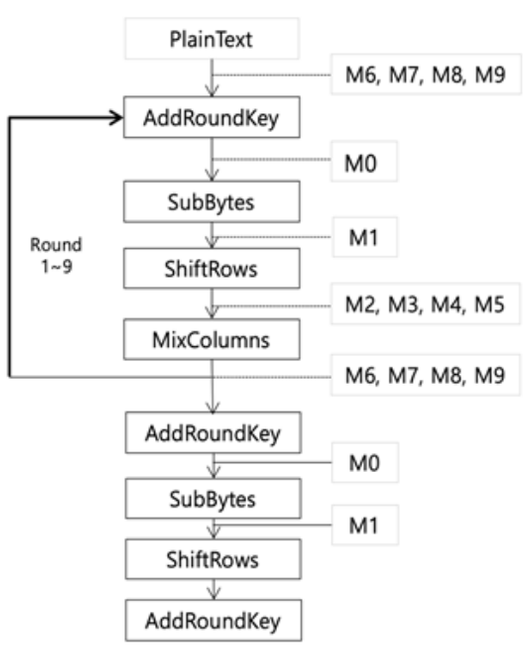


Fig. 1. Masking AES

Algorithm 1: MaskedSBOX

1: **for** i=0 and i<256
2: MaskedSBOX[i ⊕ M0] ← SBOX[i] ⊕ M1

첫 번째 라운드를 시작하기 전에 (4 × 4) 행렬의 암호화할 평문(암호문)을 첫 번째 행은 M6, 두 번째 행은 M7, 세 번째 행은 M8, 마지막 행은 M9으로 XOR 연산을 수행한다. 그리고 첫 번째 라운드부터 AddRoundKey 연산에서 모든 암호문을 M0 값으로 XOR 연산을 수행한 뒤 사전에 M6, M7, M8, M9 값을 제거하기 위해 이와 동일한 값을 XOR한다. 이 과정에서 M6, M7, M8, M9 값을 먼저 XOR 연산하게 되면 일시적으로 암호문이 그대로 노출되는 취약점이 생길 수 있기 때문에 M0 값의 XOR 연산을 수행한 이후에 XOR 연산을 하는 것이 안전하다. 그 이후 M0 값으로 마스킹된 암호문은 SubBytes 연산에서 사전에 만든 마스킹 된 Sbox 테이블을 참조하면서 M0 값이 제거되고 M1 값으로 마스킹 된다.

ShiftRows 연산에선 특별한 마스킹 연산이 이루어지지 않고 MixColumns 연산이 이루어지기 전에 모든 암호문에 첫 번째 행은 M2, 두 번째 행은 M3, 세 번째 행은 M4, 마지막 행은 M5 값으로 다시 마스킹하고 사전에 마스킹 되어있는 M1 값을 제거한다. 마지막으로 MixColumns 연산 과정에서 M2, M3, M4, M5 값이 자연스럽게 M6, M7, M8, M9으로 변경된다. 이와 같이 AddRoundkey - SubBytes - ShiftRows - MixColumns의 연산 과정을 9번의 라운드 동안 반복하기 때문에 9번의 라운드가 끝난 뒤에는 M6, M7, M8, M9의 값이 마스킹 된 암호문이 생성된다.

마지막 10라운드에서 AddRoundkey - SubBytes - ShiftRows 연산을 통해 암호화 과정을 마무리하면 암호문에 M1 값이 마스킹 값으로 적용된다. 마지막으로 이전에 사용하던 AddRoundKey 연산에서 M6, M7, M8, M9 마스킹 연산을 포함하지 않는 AddRoundKey 연산을 이용하여 M1 값을 제거하면서 모든 마스킹 연산이 제거된 암호문을 얻을 수 있다.

결과적으로 마스킹 값은 각 라운드마다 암호화 연산 과정에서 같은 값과 XOR 연산이 이루어지기 때문에 자연스럽게 사라지며 AES 알고리즘의 결과 값인 암호문은 마스킹을 적용하지 않은 암호문과 동일하지만 암호화 과정에서 필요 없는 연산들을 추가하여 전력 소모량 측정을 방해하는 효과를 얻을 수 있다.

3. 최적화 구현 기법

본 장에서는 기존의 AES 알고리즘의 연산 과정을 통합하여 오버헤드를 최소화하고 추가적인 마스킹 연산으로 인한 오버헤드를 최소화하기 위해서 마스킹 연산을 효율적으로 배치한다. 마지막으로 반복되는 라운드 함수를 Inline Assembly를 활용하여 제작함으로써 컴파일 과정에서의 비효율적인 연산을 최소화하여 오버헤드를 줄일 수 있는 소프트웨어 최적화 구현을 제시한다.

3.1 사전 Key 연산

AES 알고리즘은 라운드 별로 사용하는 키 값이 다르기 때문에 라운드 마다 이전 라운드에서 사용한 키 값을 이용하여 다음 라운드에 사용할 키 값을 생성한다. 이 과정은 반복되는 라운드 마다 새로운 키 값을 계산을 해야 하는 오버헤드를 가지는데 해당 연산을 AES 알고리즘이 최초 시작할 때 한번의 연산 과정으로 실제 키 값을 포함하여 총 11세트의 키를 생성하여 오버헤드를 줄일 수 있다.

또한 AES 암호화 과정에서 키 값이 사용되는 연산은 AddRoundKey 연산 밖에 없다. 따라서 해당 연산에서 사용되는 마스킹 값인 M6, M7, M8, M9 값과 M0 값을 암호화 초기에 라운드 키 값을 생성할 때 XOR 연산을 미리 추가하여 라운드 마다 마스킹 연산을 추가해야 하는 오버헤드를 줄일 수 있는 장점이 있다.

사전 Key 연산의 경우 라운드 키를 메모리에 미리 저장하기 때문에 기존의 구현 방법과 비교했을 때 메모리 사용량이 증가하는 단점을 가지고 있다. 하지만 일정 크기의 메모리를 사용하여 Clock Cycle을 효과적으로 감소시킬 수 있기 때문에 암호화를 실행하는 디바이스에 메모리 여유 공간이 있다면 충분히 고려할만 하다.

3.2 연산 과정 통합

최적화된 AES를 도식화하면 Fig. 2와 같다. AddRoundkey - SubBytes - ShiftRows - MixColumns 연산을 반복적으로 수행하는 AES 알고리즘을 보다 효율적으로 계산하기 위해서 AddRoundKey 연산과 SubBytes 연산을 하나로 합쳐서 구현하였다. 그리고 SubBytes연산이 끝난 뒤 M2, M3, M4, M5 값을 마스킹 연산해야 하는데 이 과정 또한 합쳐진 연산 과정에 바로 적용함으로써 한 번의 분기문과 메모리에 두 번 접근해야 하는 오버헤드를 줄일 수 있다.

그리고 ShiftRows 연산의 경우 배열에 저장된 값의 순서를 바꿔야 하는 Swap 연산을 수행한다. Swap 연산은 추가적인 메모리를 이용해야 하고 저장 순서를 바꾸는 과정에서 메모리 접근이 많은 오버헤드가 큰 연산이다.

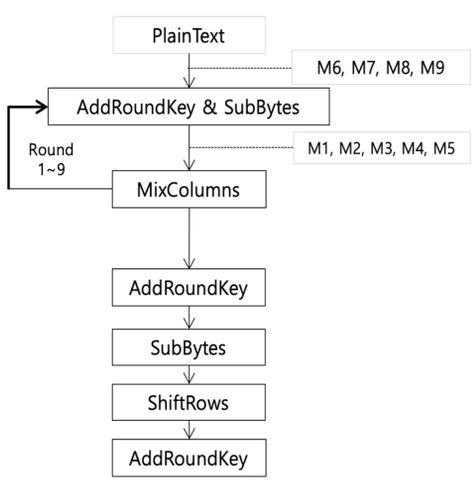


Fig. 2. Optimized AES

따라서 Swap 연산을 하지 않고 MixColumns 연산에서 배열의 인덱스를 이용하여 ShiftRows 연산이 된 것처럼 연산을 수행하여 명령어 수와 연산 시간을 효과적으로 줄일 수 있는 장점이 있다.

3.3 Inline Assembly 구현

C언어와 같은 고급 언어로 암호화 알고리즘을 구현할 경우 컴파일 과정에서 불필요한 연산이 의도치 않게 들어가는 경우가 있다. 이러한 불필요한 연산은 연산 시간을 늘리고 명령어 수 또한 늘리기 때문에 성능에 민감한 암호화 알고리즘은 Assembly 언어로 작성하는 것이 최적화 구현에 적합하다.

그리고 메모리 액세스 속도는 CPU에서 멀어질수록 상당한 시간 차이가 발생하는데 Assembly를 이용하여 자주 사용하는 값을 레지스터에 지속적으로 저장하여 사용함으로써 메모리에 액세스하는 시간을 줄일 수 있다.

Assembly 언어 같은 경우 사용하는 하드웨어에 종속되기 때문에 하드웨어의 특성을 잘 파악해야 한다. 본 논문에서 사용하는 Atmel 사의 Atmega128 보드 같은 경우 32개의 8bit 레지스터를 사용하여 연산 및 데이터 Load 및 Store 할 수 있고 16bit 주소 체계를 사용하기 때문에 정해진 8bit 레지스터 2개를 같이 사용하여 16bit 포인터 연산을 할 수 있다.

AES 알고리즘의 경우 대부분의 연산 시간을 라운드 함수에서 사용함으로 해당 부분을 Inline Assembly를 이용하여 레지스터에 비해 액세스 속도가 느린 메모리 접근을 최소화하고 대부분의 연산을 레지스터를 이용함으로써 속도를 향상시킬 수 있다. 또한 컴파일 과정에서 생기는 불필요한 명령어 코드들을 제거하여 명령어 코드 수를 줄일 수 있는 장점이 있다.

본 논문에서는 AES 암호화 알고리즘에서 가장 많은 Clock Cycle을 차지하는 10번의 라운드를 반복하는 라운드 함수를 Assembly로 구현하였다. 라운드 함수의 경우 라운드 키 로드와 Sbox 연산 등 메모리 로드 및 저장 연산이 반복적으로 수행되는 것을 확인할 수 있는데 이러한 연산을 반복문을 사용하지 않고 32개의 레지스터를 모두 이용해서 메모리 접근을 최소화하여 Clock Cycle을 효과적으로 줄일 수 있다.

4. 성능 분석

4.1 Clock Cycle

본 논문에서 제시하는 최적화 방식의 Clock Cycle 및 명령어 수는 Atmel Studio의 Atmega128 시뮬레이터를 이용하여 측정하였으며 직접 C언어로 구현한 일반적인 AES 알고리즘 및 1차 마스킹이 적용된 AES 알고리즘 성능과 비교한다.

마스킹 연산을 포함하지 않는 AES 알고리즘, 마스킹 연산을 포함한 AES 알고리즘 그리고 마스킹 연산을 포함한 최적화된 AES 알고리즘의 명령어 수 및 실행 시간을 키 길이에 따라 비교하고 1차 CPA 공격을 시도하여 최적화된 1차 마스킹 AES 알고리즘이 CPA 공격으로부터 데이터를 보호하고 암호화 속도 또한 C언어로 구현된 AES 알고리즘 보다 효율적임을 증명한다.

Table 1. Round Function Clock Cycle by Encryption Method

	AES	Masked AES	Assembly Masked AES
AES-128	6383	6829	5132
AES-192	7734	8102	6201
AES-256	9576	10859	7197

Table 1은 C언어로 구현된 일반적인 AES-128, AES-192, AES-256 알고리즘과 마스킹 연산이 포함된 Masked AES 그리고 Assembly로 구현된 Masked AES 알고리즘의 최적화 옵션 레벨 3으로 컴파일 된 라운드 함수의 Clock Cycle을 나타낸다. C언어로 구현된 Masked AES 알고리즘에 비해 Assembly로 구현된 AES 알고리즘이 훨씬 적은 Clock Cycle을 보이는 것을 볼 수 있다.

또한 키의 길이가 길어질수록 더 많은 라운드를 돌기 때문에 클록수가 큰 것을 볼 수 있는데 최적화 과정을 통해서 그 차이를 줄인 것을 확인할 수 있다. 키 길이가 256bit인 경우 일반적인 AES에 비하여 Assembly AES의 클록수가 크게 낮아진 것을 확인할 수 있고 C 언어로 제작된 Masked AES 에 비하여 대략 20%의 성능 향상이 이루어진 것을 볼 수 있다.

4.2 전력 분석 공격에 대한 안전성

구현한 AES가 전력분석 공격에 안전함을 확인하기 위하여 ATxmega128 MCU에서 암호화 연산을 수행하는 동안 발생하는 소비 전력을 Chipwhisperer-Lite (CW1173)로 측정하여 실험을 진행하였다. 실험은 마스크를 적용하지 않은 AES와 본 논문에서 제시한 마스크를 적용하고 최적화한 AES와 비교한다. AES에서의 CPA 공격은 Sbox 조회가 수행되는 시점에서 암호화 장치의 전력 소비를 모니터링하는 경우 정보의 유출로 공격이 가능하다. 그렇기 때문에 SubBytes의 출력값을 중간값으로 하여 1라운드에서 공격을 5,000개의 파형의 수로 수행하였다.

또한 부채널 분석에 대응하기 위한 마스킹 연산을 수행하는 경우 동일한 레지스터에 같은 마스킹 값을 가지는 다른 값을 덮어씌우는 경우 기존에 있던 마스킹 값이 지워지는 보안 취약점이 존재할 수 있다. 이러한 취약점을 막기 위해 동일한 마스킹을 사용하는 암호문의 경우 동일한 레지스터를 사용하지 않도록 유의하여 Assembly를 구현하였다.

실험결과 Fig. 3와 Fig. 4은 파형의 수에 따라 추측되는 키의 상관계수 그래프를 나타낸다. Fig. 3에서는 마스킹이 적용되지 않은 AES-128 알고리즘에 CPA 공격을 시도하여 상관계수가 가장 높은 키 값이 두 번째로 높은 키 값에 비하여 상당히 높게 분석된 것을 확인할 수 있다. 아래의 파형을 살펴보면 빨간색으로 표시된 실제 키 값이 초록색으로 표시된 키 값이 아닌 값들과 뚜렷하게 비교되는 것을 알 수 있다. 이런 경우 공격자가 실제 키 값을 쉽게 유추할 수 있다.

그러나 Fig. 4에서는 마스킹 연산이 이루어진 AES 알고리즘에 CPA 공격을 해도 큰 상관 계수를 보인 Fig. 3에 비해서 상관 계수에 큰 차이가 없어서 특정 키 값을 찾아낼 수 없는

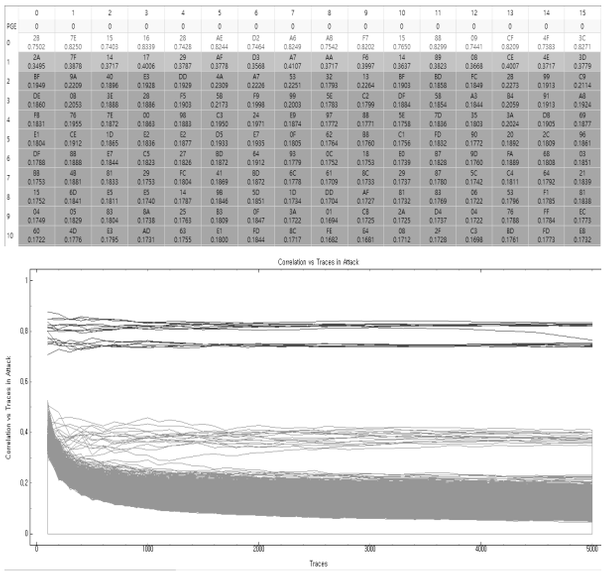


Fig. 3. AES with CPA

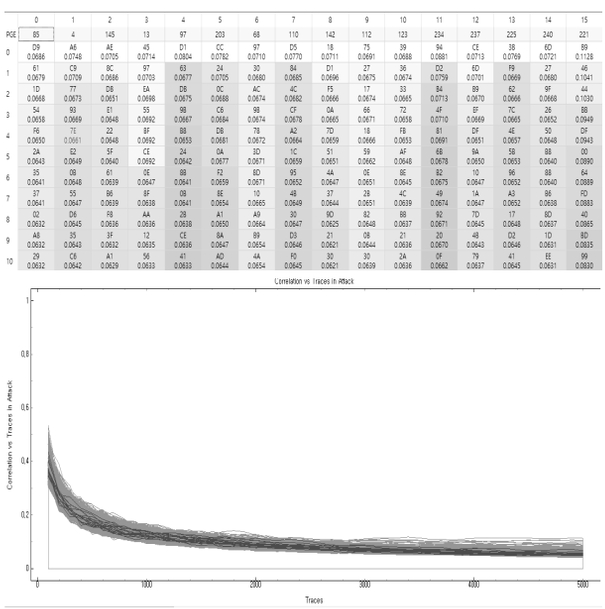


Fig. 4. Masked AES with CPA

것을 확인할 수 있다. 2번째 키의 경우 실제 키 값의 상관도가 4번째로 높게 분석되었지만 다른 예상 키 값의 상관도와 큰 차이가 없기 때문에 공격자가 실제 키 값이라고 판단하기엔 어려움이 있다. 뿐만 아니라 아래 파형을 살펴보아도 Fig. 3의 파형에 비해 다른 값들과 크게 차이나는 부분 없이 전체적으로 비슷한 모양의 파형이 반복되는 것을 확인할 수 있다. 이를 통해 구현한 1차 마스킹 AES가 전력 분석 공격에 안전함을 확인하였다.

추가적으로 AES-192와 AES-256 알고리즘의 CPA, DPA 공격의 경우 AES-128 공격 코드를 변형하여 분석한다. 따라서 본 논문에서 AES-128 알고리즘의 안전성을 확인했기 때문에 AES-192, AES-256도 안전한 것을 확인 할 수 있다.

5. 결 론

본 논문에서는 AVR 프로세스 상에서 부채널 분석 공격에 대응하기 위해 AES 암호화 알고리즘에 1차 마스킹 기법을 적용하였고 Inline Assembly를 이용한 소프트웨어 최적화 구현 기법에 대해 구현하였다. 이는 부채널 분석 공격에 취약점을 보인 기존의 AES 알고리즘을 보완하였고 Inline Assembly를 이용하여 실행 시간에서 C언어에 비해 좋은 성능을 보임을 증명하였다.

AES 알고리즘은 다른 경량 암호화 알고리즘에 비해 연산량이 많아서 비교적 많은 실행 시간이 필요하기 때문에 사물인터넷 디바이스에서 사용하기 위해서는 최적화가 필요하다. 또한 전력 소모량 분석을 이용한 부채널 공격에 대한 AES의 취약점 연구 사례가 많이 나오고 있는 만큼 마스킹을 이용한 부채널 공격 대응 및 Inline Assembly를 이용한 최적화 소프트웨어 구현은 의미가 있다.

본 논문에서 제안하는 소프트웨어 최적화 구현을 토대로 앞으로의 연구 방향은 Assembly를 이용한 효율적인 암호화 구현을 제시하는 데 있다.

References

[1] Moon Si-hoon, Kim Min-woo, and Kwon Tae-kyung, “Trends in Lightweight Crypto Technology for IoT Communication Environments,” *The Journal of The Korean Institute of Communication Sciences*, Vol.33, No.3, pp.80-86, 2016.

[2] Hwajeong Seo and Howon Kim, “Implementation of Light-weight Encryption Algorithms for the Internet of Things,“ *REVIEW OF KIISC*, Vol.25, No.2, pp.12-19, 2015

[3] P. C. Kocher, “Timing Attacks on Implementation of Diffe-Hellman, RSA, DSS and Other Systems,” *Proc. Adv. Cryptology*, pp.104-113, 1996.

[4] E. Biham and A. Shamir, “Differential Fault Analysis of Secret Key Cryptosystems,” *Proceedings of Crypto 1997, LNCS 1294*, pp.513-525, Aug. 1997.

[5] J. Quisquater and D. Samyde, “Electromagnetic Analysis (EMA): Measures and Countermeasures for Smart Cards,” P*roc. e-Smart*, pp.200-210, 2001,

[6] National Institute of Standards and Technology, “Advanced Encryption Standards,” NIST FIPS PUB 197, 2001.

[7] R. Rivest, A. Shamir, and L, Adelman, “A method for obtaining digital signature and public-key cryptosystems,” *Comm. of the ACM 21*, pp.120-126, 1978.

[8] D. Hong, J. Lee, D. Kim, D. Kwon, K. Ryu, and D. Lee, “LEA, A 128-bit block cipher for fast encryption on common processors,” *WISA'13, LNCS 8267*, pp.3-27, 2014.

[9] TTA, “128-bit lightweight block cipher LEA,” TTAK.KO-12.0223, Dec, 2013.

[10] D. Kwon, J. Kim, S. Park, S. Sung, Y. Sohn, J. Song, Y. Yeom, E. Yoon, S. Lee, J.Lee, S. Chee, D. Han, and J. Hong, *New Block Cipher : ARIA In ICISC'03, LNCS 2971*, pp.

432-445, Springer-Verlag, 2003.

[11] Y. Kim and H. Yoon, “First Experimental Result of Power Analysis Attacks on a FPGA Implementation of LEA,” IACR Cryptology ePrint Archive, 2014:999, Available at https://eprint.iacr.org/ 2014/999.pdf

[12] J. Park, T. Kim, H. An, Y. Won, and D. Han, “Side channel attacks on LEA and its countermeasures,” *Journal of The Korea Institute of Information Security & Cryptology (JKIISC)*, Vol.25, No.2, pp.449-456, 2015.

[13] JungKab Seo, ChangKyun Kim, JaeCheol Ha, SangJae Moon, and IlHwan Park, “Differential Power Analysis Attack of a Block Cipher ARIA,” *Journal of the Korea Institute of Information Security & Cryptology(Korea Institute Of Information Security And Cryptology)*, Vol.15, No.1, pp.99-107, 2005.

[14] HyungSo Yoo, JaeCheol-Ha, ChangKyun Kim, IlHwan Park, and SangJae Moon, “A Secure ARIA implementation resistant to Differential Power Attack using Random Masking Method,” *Journal of the Korea Institute of Information Security & Cryptology*, Vol.16, No.2, pp.129-139, 2006.

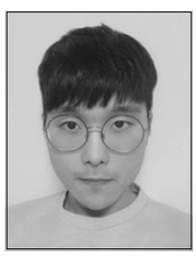
[15] ChangKyun Kim, JaeHoon Park, Daewan Han, and Dong Hoon Lee, “Investigation of Masking Based Side Channel Countermeasures for LEA,” *Journal of the Korea Institute of Information Security & Cryptology*, Vol.26, No.6, pp.1431-1441, 2016.

[16] L. Goubin and J.Patarin, “DES and Differential Power Analysis – The Duplication Method,” CHES 1999, LNCS 1717, pp.158-172, Springer, 1999

[17] Paul Kocher, Joshua Jaffe, and Benjamin Jun, “Differential Power Analysis,” CRYPTO '99, Springer-Verlag, 1999, pp.388-397.

[18] Eric Brier, Christophe Clavier, and Francis Olivier, “Correlation Power Analysis with a Leakage Model,” CHES 2004: Cryptographic Hardware and Embedded Systems - CHES 2004, pp.16-29.

[19] C. Herbst, E. Oswald, and S. Mangard, An AES Smart Card Implementation Resistant to Power Analysis Attacks. Lecture Notes in Computer Science, pp.239-252, 2006.



김 경 호

https://orcid.org/0000-0002-0945-3837

e-mail : pgm.kkh@gmail.com

2019년 한성대학교 IT응용시스템공학과 (공학학사)

2019년~현 재 한성대학교 IT융합공학과 석사과정

관심분야 : 시스템 보안, 암호화 구현

서 화 정

https://orcid.org/0000-0003-0069-9061

e-mail : hwajeong84@gmail.com

2010년 부산대학교 컴퓨터공학과(학사)

2012년 부산대학교 컴퓨터공학과(석사)

2012년~2016년 부산대학교 컴퓨터공학과 (박사)

2016년~2017년 싱가포르 과학기술청

2017년~현 재 한성대학교 IT융합공학부 조교수

관심분야 : 정보보호, 암호화 구현, IoT