단일 부채널 전력 파형을 사용한 마이크로컨트롤러 상에서 소프트웨어 표절 탐지

김현준*, 장경배*, 김경호*, 서화정*
*한성대학교 IT융합공학부
khj93072004@gmail.com

Similar Software Code Detection Using Side Channel Leakage in Microcontrollers

Hyun-Jun Kim*, Kyung-Bae Jang*, Kyung-Ho Kim*, Hwa-Jeong Seo*
*Divison of IT convergence engineering, Hansung University

요 약

부채널 정보을 사용하여 마이크로 컨트롤러 상에서 표절 된 코드를 탐지하는 새로운 방법을 제시한다. 제안 기법은 애플리케이션을 보호하기 위해 추가로 워터 마킹 할 필요가 없이 코드를 실행하는 마이크로 컨트롤러의 유출데이터를 워터 마크로서 사용 할 수 있다. 두 가지 다른 구현의 각각 하나의 부채널 파형에 대한 절대 상관 계수를 기반으로 분석 한다. 어셈블리 언어로 작성된 다양한 테스트 응용 프로그램을 사용 Xmaga128 마이크로 컨트롤러에서 평가하였다. 제안 기법은 어셈블리 코드를 수정하는 공격자에게도 강력하며 코드에 대한 정보와 입력에 대한 접근이 불가능 하여도 탐지가 가능하다.

1. 서론

임베디드 시스템에서 작동하는 소프트웨어는 지적 재산권으로 보호받을 수 있으며 일반적으로 개발자가 지적 재산권 (IP) 코어라고 하는 타사 디자인을 구매하고 사용한다. Marketsandmarkets의 연구에 따르면 임베디드 시스템은 2020년 865억 달러에서 2025년 1,110억 달러로 성장할 것으로 예상된다. 2020 년에서 2025년까지 연평균 6.1 % 증가할 것으로 예상하고 있다[1]. 이와 함께 IP 코어 판매에 대한 시장도 커지며 IP 도용 또한 중요한 위협이 되었다.

IP는 리버스 엔지니어링을 통해 해킹하고 디자인을 불법 복제하여 판매하거나 사용 할 수 있다. 이런 문제는 제품 수익에 큰 영향을 끼치는 만큼 임베디드 시스템 제조업체의 큰 관심 사항이며 IP 불법 복제를 방지하기 위해 다양한 기술이 제안되었다.

이러한 IP 불법 복제를 방지하기 위한 방법으로 워터마킹 기술을 활용하는 방법이 있다. 소프트웨어 워터마크는 공격자가 프로그램을 복사하는 것을 막지는 않지만, 복제를 탐지하는 데 사용된다. 워터 마킹은 리버스 엔지니어링 분석보다 저렴한 비용으로 불법 복제물을 탐지할 수 있다.

워터마크 기법에는 감지하는 소프트웨어나 실행 중인 메모리에 대한 접근이 필요하다. 그러나 코드 복사가 관련 위협이 있는 시스템의 경우에서는 대부분 메모리가 무단 읽기 작업으로부터 보호되기 때문에 임베디드 환경에서는 의심스러운 코드에 접근하기가 쉽지 않다. 이 문제를 해결하기 위해 최근 연구에는 프로그램 코드에 접근하지 않고 감지 할 수 있는 부 채널 정보 기술을 사용한 워터마킹 방법이 제안되고 있다.

관련된 가장 최근의 연구에서는 공격자가 IP 표절 탐지를 막기 위해 마이크로 컨트롤러 상에서 코드가 작동할 때 나타나는 전력 파형을 비교하여 표절 여부를 판단한다[2, 3, 4, 5]. 이때 분석을 위해 10만 개 이상의 파형과 같은 입력값을 사용하나 입력이 알려지지 않거나 입력 데이터를 조작할 수 없는 경우에는 이러한 분석이 불가능하다[5]. 본 논문에서는 다른 입력값에서도 적용되며 하나의 파형으로도 분석 가능한 새로운 기법을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 제안하는 기법에 대해 설명한다. 3장에서는 제안 기법에 대한 실험을 통해 성능을 평가한다. 4장에서는 본 논문의 결론과 추후 연구 방향에 대해 기술한다.

2. 제안기법

kocher의 부채널 공격에 대한 개척 이후 마이크롤러 컨트롤러에서 코드가 실행되는 동안 전력 소비,

전자기 방출에서 사이드 채널 누설을 유발한 다는 것이 잘 알려졌다[6]. 마이크로 컨트롤러에서 수행되는 명령어들은 각각의 다른 전력 소비를 갖는다. 제한 기법에서는 IP 보호를 구현된 프로그램 코드와 구현된 미지의 의심스러운 코드의 전력 파형을 사용한다. 두 코드의 파형의 상관 계수를 계산하면 두 코드의 유사도가 높을수록 상관계수가 높게 나타난다. 만약 공격자가 코드를 복사 할 경우 코드 상에서 동일한 부분은 높은 상관계수를 나타낼 것이다. 제안 기법에서는 원본파형을 일정길이의 구간으로 나누어 원본파형이 드러나는 부분을 탐색하여 IP 탐지에 적용하였다.

공격 공격은 두 가지의 경우로 나눌 수 있다. 첫 번째로는 공격자가 도난한 IP를 그대로사용하는 경우이다. 두 번째는 공격자가 도난한 IP를 수정하여 사용하는 방법이다. 첫 번째의 경우는 일반적인 워터마킹의 기법으로 IP복제의 여부를 판단 할 수 있지만 두 번째의 경우에는 워터마킹으로 사용된 코드 부분을 수정하여 IP탐지를 방해 할 수 있다. 제한 기법에서는 공격자가 이 두 가지의 공격에 대한 방어를 중점으로 두었다.

추출 데이터의 추출은 정품의 프로그램과, 알려지지 않은 프로그램의 파형을 수집하여야한다. 정품 코드의 파형을 워터 마킹으로 사용하며 대상장치에서 사용되는 입력값을 선택 할 수 없다고 가정하에 해당코드는 알려지지 않은 입력에 대한 입력을 사용한다. 측정은 유사성을 비교하기 쉽도록 두 프로그램은 동일한 측정 설정을 사용한다.

전처리 동일한 파형이라도 노이즈로 인해 변경될 수 있다[4]. 데이터의 압축과 노이즈에 대한 필터링을 위해 FFT (Fast Fourier Transform)를 적용한다.

탐지 탐지는 추출한 두 개의 데이터에서 4개의 행렬을 생성한다. 이 4개의 행렬을 사용하여 상관계수를 계산하고 최대값으로 사영하고 이 값을 비교한다.

추출 데이터 기반으로 행렬 생성 추출한 정품코드의 파형 데이터를 N의 길이로 나눈 N x M1 행렬 $T_{genuine}$과 추출한 의심코드의 파형 데이터를 N의 길이로 나눈 N x M2 행렬 $T_{unknown}$을 생성한다. M은 각 데이터 샘플의 길이를 N으로 나눈 값이다.

추출한 코드의 파형 데이터의 길이 l을 N만큼 추출한 값 을 모든 구간에서 추출하여 FFT 적용

$$T' = \begin{pmatrix} a_0 & a_1 & \cdots & a_{n-2} & a_{n-1} \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ a_{l-n} & a_{l-n+1} & \cdots & a_{l-n+1} & a_{l-1} \end{pmatrix}$$

를 생성한다. 정품 파형의 데이터로 N x M1‘ 행렬 $T'_{genuine}$과 의심 파형의 데이터로 N x M2‘ 행렬 $T'_{unknown}$을 만든다.

피어슨 상관계수 계산 제안기법에서는 전력 기반 암호 분석 분야에서 많이 사용되고 있는 피어슨 상관관계를 사용한다. 정품 코드와 미지의 코드에 데이터 누출 간의 유사성을 계산 하기 위해 행렬 $T_{genuine}$와 $T'_{unknown}$의 상관계수로 이루어진 M1 x M2‘ 행렬 S1 그리고 행렬 Tu_{nknown} 와 $T'_{genuine}$의 상관계수로 이루어진 M2 x M1‘ 행렬 S2를 구한다.

Maximum Projection 구간으로 나눈 파형이 비교하는 파형과 유사한 지점이 있다면 해당 구간은 그 지점에서 높은 상관계수를 갖는다. 이러한 값을 찾기 위해 S1와 S2의 열에서 최대값으로 이루어진 $p1_{col}$와 $p2_{col}$를 구한다.

이 $p1_{col}$와 $p2_{col}$의 평균치로 표절 여부를 판단하고 각 구간에 대한 값을 사용하여 해당 구간에 대한 유사성을 판단하여 표절 후 수정 여부를 평가한다.

3. 실험

실험에서 장치의 수집은 입력이 알려지지 않은 단일 파형이 사용되며 소스 코드 또한 알려지지 않았다. 측정에는 동일하게 ChipWhispererLite XMEGA (8-bit processor)[7] 사용하며 7.38 MS/s 샘플링으로 모두 동일한 설정을 사용한다.

3.1 같은 암호에 다른 구현 비교

같은 알고리즘에도 구현에 따라 속도와 구조가 다르다. IP의 구분에 있어서도 서로 다른 IP로 간주 해야한다. 또한 코드의 일부가 유사하거나 동일하다는 것을 확인하기 위해 3가지의 AES 암호화 구현에 대한 테스트를 진행한다.

<표 1> Furious와 다른 AES암호구현과의 제안기법을 사용한 n에 따른 절대 상관계수

	400	200	100	50
Furious	0.99213	0.99170	0.99287	0.99486
	0.99216	0.99194	0.99320	0.99507
Fast	0.83707	0.88271	0.92129	0.96302
	0.88861	0.92829	0.95482	0.98099
Fantastic	0.84027	0.88468	0.91259	0.94188
	0.88577	0.91963	0.94888	0.96943

<표 1> 은 모든 테스트 AES 구현에 대한 평균 절대 상관 계수를 보여준다. 라운드의 반복으로 같은 구간에 대한 비교로 유사한 파형이 반복됨을 알 수 있다. (그림 1)과 같이 같은 구현을 비교할 경우 N의 값이 작아지더라도 0.99 이상의 비교적 높은 상관계수를 나타낸다.

다른 구현의 경우에는 N의 값이 클수록 높은 상관계수를 나타내며 N의 작을 경우 (그림 2)와 같이 동일한 코드의 부분에서는 높은 상관계수를 나타내고 그렇지 않은 부분에서는 낮은 상관계수를 드러낸다. 그래프를 통해 코드의 어느 부분이 유사하거나 동일한 지 알 수 있다.

3.2 표절 후 수정된 구현 비교

코드를 적극적으로 조작하는 공격자를 시뮬레이션하기 위해 Furious를 실제 구현으로 선택하고 6개의 방식으로 수정한다.

주소변경 구현에서 사용되는 레지스터와 SRAM 주소를 변경한다.

명령어변경 가능한 경우 클럭 사이클의 변경 없이 명령의 순서나 명령어를 바꾼다.

주소와 명령어 변경 구현에서 사용되는 레지스터와 SRAM 주소 그리고 명령의 순서를 바꾼다.

NOP 더미 추가 더미로 NOP명령어를 삽입한다. 실험에는 570 사이클을 추가 하였다.

Smart 더미 추가 더미로 NOP 명령어를 사용시에는 전력 소비가 적기 때문에 구별이 쉽다. 전력소비를 보다 정교하게 보이기 위해 Smart 더미[5]를 사용한다. 실험에는 570 사이클을 추가 하였다.

주소와 명령어 변경과 Smart 더미 추가 레지스터와 SRAM 주소 그리고 명령의 순서를 변경하고 Smart 더미[5]를 추가한다.

<표 2> 실제 구현과 수정한 코드의 제안기법을 사용한 n에 따른 절대 상관계수

	400	200	100	50
addr	0.98174	0.98092	0.98204	0.98700
	0.98230	0.98124	0.98272	0.98786
swap	0.99026	0.98946	0.98998	0.99198
	0.99023	0.98943	0.99021	0.99218
addr+swap	0.98123	0.98073	0.98243	0.98707
	0.98168	0.98176	0.98338	0.98773
dummy (nop)	0.88323	0.92220	0.93959	0.96058
	0.88191	0.92733	0.96737	0.98681
dummy (smart)	0.78183	0.85451	0.91320	0.95578
	0.81031	0.91686	0.96328	0.98745
addr+swap+ dummy(smart)	0.79846	0.85621	0.91140	0.95135
	0.81913	0.91282	0.95799	0.98227

<표 2>는 수정한 코드의 평균 절대 상관 계수를 보여준다. 레지스터 수정과 명령어 교환에서는 수정과 무관하게 높은 평균의 상관계수를 나타내 표절하여도 탐지가 가능하다.

더미를 추가 하였을 경우에는 더미의 추가가 상관계수를 낮추는 것을 알 수 있다. 그러나 더미 추가의 경우 N의 크기를 줄이면 (그림3), (그림4)와 같이 수정 전 코드구간을 수정코드에 탐색할 경우에는 전체적으로 높은 상관계수를 나타내지만 반대의 경우는 더미를 추가한 구간의 상관계수가 낮게 나와 이를 구분할 수 있다. 그림 과 같이 유사성을 나타내는 그래프에서 차이로 더미를 추가하였음을 알 수 있어 더미 코드를 추가 하였어도 탐지가 가능함을 확인할 수 있다.

4. 결론

마이크로 컨트롤러 플랫폼에서 소프트웨어 표절을 탐지하기 위한 새로운 방법을 제시하고 평가하였다. 소스 코드와 입력이 데이터가 알려지지 않은 까다로운 시나리오에서 IP를 식별 할 수 있었다. 또한 실험에서 제안하는 방법이 여러 가지 코드 변환에 강력함하며 코드에 추가적인 작업이 필요 없이 적은양의 마이크로 컨트롤러의 데이터 유출만으로 IP 식별이 가능함을 보였다

참고문헌

[1] Embedded System Market by Hardware, Software, System Size, Functionality, Application, Region - Global Forecast to 2025. https://www.marketsandmarkets.com/Market-Reports/embedded-system-market-98154672.html

[2] Becker, G., Strobel, D., Paar, C., Burleson, W.: Detecting software theft in embedded systems: a side-channel approach. IEEE Trans. (2012)

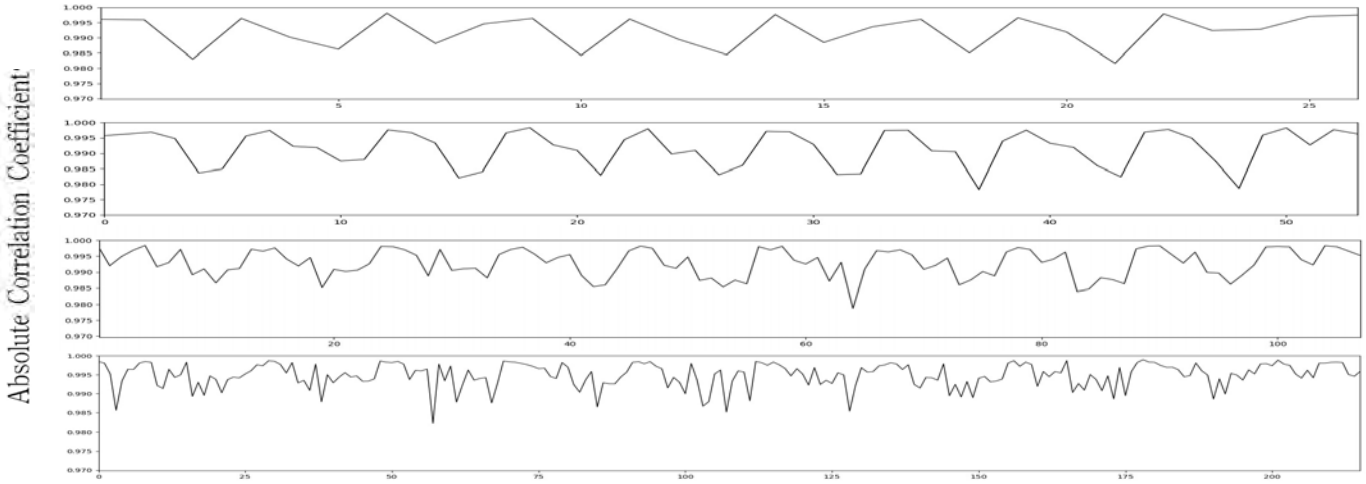
[3] Strobel, D., Bache, F., Oswald, D., Schellenberg, F., Paar, C.: SCANDALee: a sideChANnel-based DisAssembLer using local electromagnetic emanations. In: Design, Automation, and Test in Europe (DATE), 9 - 13 March 2015 (2015)

[4] Durvaux, F., G´erard, B., Kerckhof, S., Koeune, F., Standaert, F.-X.: Intellectual property protection for integrated systems using soft physical hash functions. In: Lee, D.H., Yung, M. (eds.) WISA 2012. (2012).

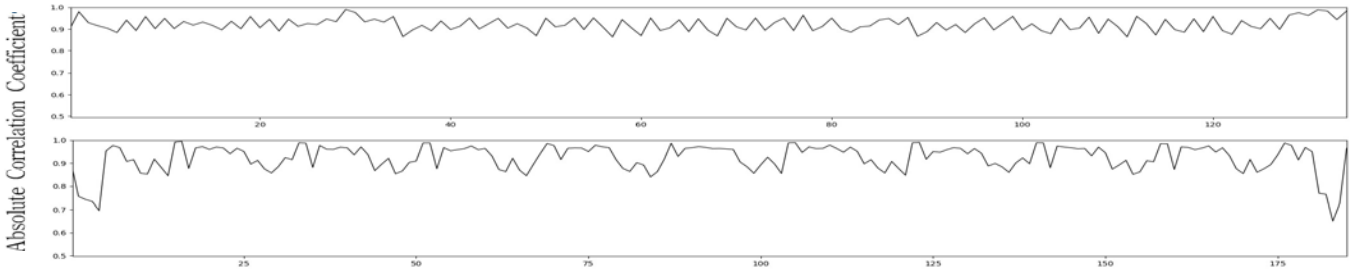
[5] Peter Samarin, Kerstin Lemke-Rust: Detecting Similar Code Segments Through Side Channel Leakage in Microcontrollers. ICISC 2017: 155-174

[6] Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. (1999).

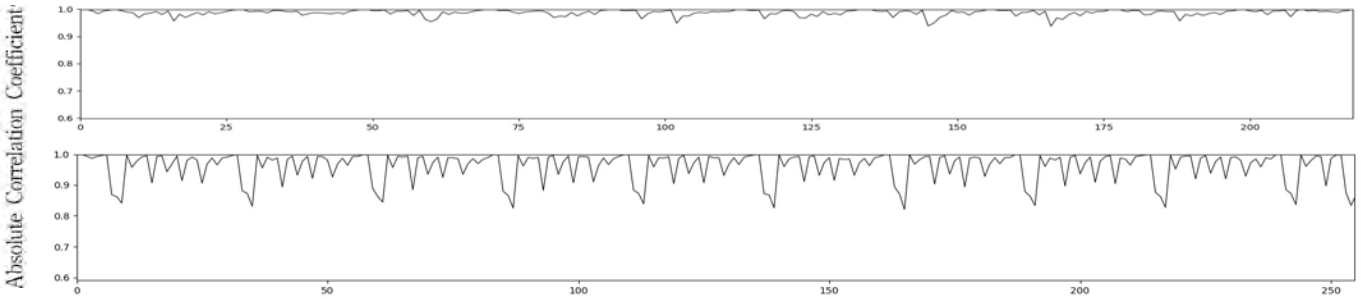
[7] NewAE Techonology Inc. ChipWhisperer-Lite. https://wiki.newae.com/ CW1173_ChipWhisperer-Lite.



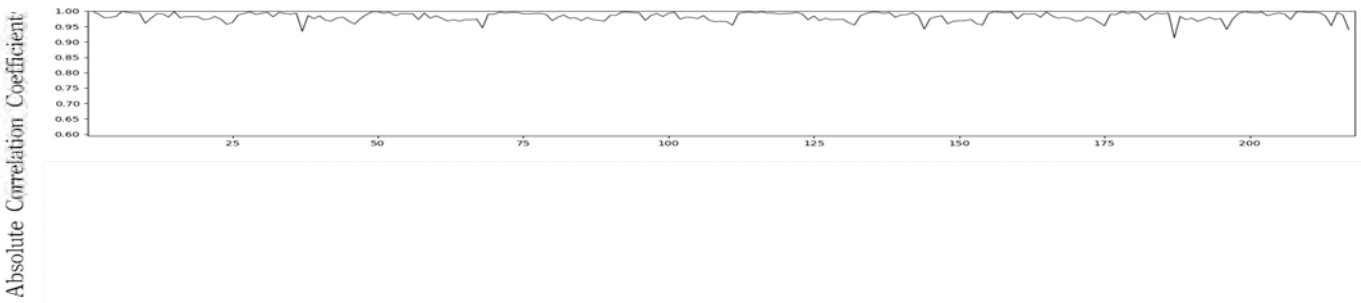
(그림 1) Furious와 Furious의 n=400, 200, 100, 50에서 p1col 그래프 n의 개수와 상관없이 높은 상관계수를 나타낸다.



(그림 2) 위 Furious와 Fast의 n=100에서 p1col 그래프와 아래 AES암호 구현 Furious와 Fantastic의 n=100에서 p1col 그래프 Furious와 Furious 비교 보다 낮은 상관계수를 나타낸다.



(그림 3) Furious와 dummy(nop)을 추가한 Furious의 n=50에서 p1col 그래프와 p2col 그래프. p1col 그래프와 다르게 p2col 그래프는 추가된 더미 코드 부분에서의 상관계수가 높게 나오는 것을 확인 가능하다.



(그림 4) Furious와 Dummy(smart)을 추가한 Furious의 n=50에서 p1col 그래프와 p2col 그래프. p1col 그래프와 다르게 p2col 그래프는 추가된 더미 코드 부분에서의 상관계수가 높게 나오는 것을 확인 가능하다.