경량 환경의 난수발생기에서도 사용 가능한 지터 엔트로피 소스의 설계에 관한 연구

김예원[2], 강주성[1,2], 염용진[1,2]*

국민대학교 정보보안암호수학과[1] / 금융정보보안학과[2]

{fdt150, jskang, *salt}@kookmin.ac.kr

A Study on the Design of Jitter Entropy Source suitable for Random Number Generators in Lightweight Environment

Yewon Kim[2], Ju-Sung Kang[1,2], Yongjin Yeom[1,2]*

Dept. of Information Security, Cryptology, and Mathematics[1] /

Financial information security[2], Kookmin Univ.

요 약

암호 시스템의 안전성을 위해 사용되는 난수발생기의 입력으로 하드웨어 난수발생기의 출력 난수, 사용자 입력 등이 사용되고 있으나, 경량 환경의 난수발생기에서는 사용할 수 없을 가능성이 크다. CPU의 타이머로 측정한 명령어 실행 시간의 변동성을 기반으로 한 지터 엔트로피 소스는 모든 환경의 난수발생기에서 사용될 가능성이 크다. 지터 엔트로피 소스는 Linux 운영체제와 Google이 새로 개발하고 있는 Fuchsia 운영체제의 난수발생기에서 사용되고 있으며, 암호모듈 검증제도에 사용되는 국내 표준문서의 엔트로피 소스 목록에 포함되어 있다. 하지만, 지터 엔트로피 소스의 설계 방법에 관한 연구는 미미하다. 따라서 본 논문은 CPU의 타이머 사이에 수행되는 명령어에 따른 지터 엔트로피 소스의 분포와 최소 엔트로피를 분석하여 지터 엔트로피 소스의 설계 방법을 제안한다. 본 연구 결과는 암호 시스템 및 암호모듈 개발자들이 경량 환경에서도 엔트로피 소스를 수집하여 암호학적으로 안전한 난수를 출력하는 데에 활용될 것으로 기대한다.

Ⅰ. 서 론

난수발생기(Random Number Generator, RNG)는 암호키, 논스(nonce), 솔트(salt), 보안 매개변수 등을 구성하는 필수 요소인 난수를 생성하므로, 현대 암호 시스템과 암호 모듈의 운용에 필수적인 요소이다. 난수발생기는 엔트로피 소스(Entropy source)를 입력받아 결정론적 알고리즘을 통해 난수를 출력하므로, 출력 난수가 입력인 잡음원에 의존적이라는 특징을 가지고 있다. 충분한 엔트로피를 수집하지 못하면 출력 난수가 예측 가능해지며, 보안 매개변수 등의 안전성을 약화하여 암호 시스템 전체의 안전성에 영향이 미치게 된다. 따라서 난수발생기가 입력으로 사용하는 엔트로피 소스의 엔트로피에 대한 분석이 중요하다.

Linux, Windows, Fuchsia 등의 여러 운영체제의 난수발생기는 Intel RDRAND와 같은 하드웨어 난수발생기의 출력 난수, 마우스/키보드 움직임 등의 사용자 입력값, 디스크 타이밍(Disk timings), 인터럽트 타이밍(Interrupt timings) 등을 엔트로피 소스로 사용한다. 이때, 한 가지 종류의 엔트로피 소스만을 사용하지 않는다. 이는 기계적 또는 환경적 변화로 인해 엔트로피 소스의 엔트로피가 충분하지 않아져 운영체제의 난수발생기가 암호학적으로 안전한 난수를 생성하지 못할 수 있기 때문이다. 반면, IoT 기기 등의 경량 환경에는 하드웨어 난수발생기와 사용자 입력이 없을 가능성이 크고 디스크 타이밍 등의 엔트로피 소스도 사용하지 못할 수 있다. 따라서 PC뿐만 아니라 모든 경량 환경에서도 사용할 수 있는 엔트로피 소스를 설계하는 연구가 필요하다.

지터(Jitter) 엔트로피 소스는 CPU의 고해상도 타이머(Timer)를 이용하여 측정한 명령어 실행 시간의 변동성을 기반으로 한다. Müller는 160번의 메모리 접근과 1번의 LFSR의 실행 시간의 변동성을 기반으로 하는 지터 엔트로피 소스를 제안하였다[1]. Müller의 지터 엔트로피 소스는 Linux의 커널 모듈(Kernel module)로 사용되고 있으며, 2016년부터 Google이 새로 개발하고 있는 Fuchsia 운영체제 및 Zircon 커널의 난수발생기에서 주요 엔트로피 소스로 사용되고 있다. 하지만, Müller의 지터 엔트로피 소스의 설계 방법에 대한 타당성을 분석한 연구는 미미하다. 국내 암호모듈 검증 제도에 사용되는 국내 표준문서는 지터 엔트로피 소스가 포함된 Linux 및 Windows 운영체제에서 사용할 수 있는 엔트로피 소스 목록을 제시한다[2]. 또한, 수집 방법에서 두 번의 고해상도 타이머 사이에 수행될 수 있는 명령어는 사용자의 정의에 따라 사용하도록 작성되어 있다[2].

본 논문에서는 IoT 기기 등의 경량 환경에서도 사용할 수 있는 지터 엔트로피 소스를 설계하는 방법에 관해 연구를 진행하였다. 먼저 시간 측정 함수의 실행 시간에 대한 변동성을 기반으로 하는 지터 엔트로피 소스에 대한 분포와 엔트로피를 분석하였다. 또한, 고해상도 타이머 사이의 명령어로 기본 연산자 또는 메모리 접근을 사용하였을 때의 지터 엔트로피 소스에 대한 분포와 엔트로피를 분석하였다. 본 논문에서는 Khadas VIM2 보드에서 Zircon 커널이 부팅되는 초기에 수집한 지터 엔트로피 소스를 분석에 사용하였다. Khadas VIM2 보드는 옥타 코어(Octa-core) ARM Cortex-A53 기반 SoC인 Amlogic S912를 사용한다.

Ⅱ. 제안하는 지터 엔트로피 소스의 설계 방법

2.1 Zircon 커널의 시간 측정 함수에 대한 지터 엔트로피 소스

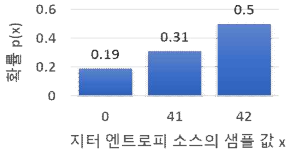
Zircon 커널의 시간 측정 함수인 current_time() 함수는 물리적 카운트(Physical counter) 값에 대한 64-bit 크기의 나노초(Nacnosecond, ns)를 출력한다. 즉, [그림 1]에서 보이는 바와 같이, 물리적 카운트에 대한 레지스터(Register)를 읽어와 1 카운트 당 나노초를 곱하는 연산을 진행한다. 측정된 ARM의 generic 타이머 주파수(Frequency)는 24 MHz이므로, 1 카운트 당 나노초는 약 41.6 ns로 계산된다.

[그림 1] Zircon 커널의 시간 측정 함수 current_time()의 동작 과정

시간 측정 함수에 대한 지터 엔트로피 소스의 샘플을 아래와 같이 구성하였다. 이때, 샘플의 크기는 64-bit이다.

```
uint64_t  time1 = current_time(); // 시간 측정 함수 호출
uint64_t  time2 = current_time(); // 시간 측정 함수 호출
uint64_t  jitter = time2 - time1; // 지터 엔트로피 소스의 샘플
```

Zircon 커널의 부팅 초기에 100만 개의 샘플을 수집하여 [그림 2]와 같은 분포를 도출하였다. Shannon 엔트로피와 최소 엔트로피(Min-entropy)를 계산하였고, 엔트로피 평가 방법에 대한 주요 표준문서인 미국 NIST의 SP 800-90B[3]의 평가법으로 최소 엔트로피를 추정하였다. SP 800-90B의 평가법에서 지터 엔트로피 소스는 Non-IID(Independent Identically Distributed)로 판정되었다. 시간 측정 함수에 대한 지터 엔트로피 소스의 엔트로피(①)는 세 가지의 추정된 엔트로피에서 최솟값인 0.19-bit로 여긴다.

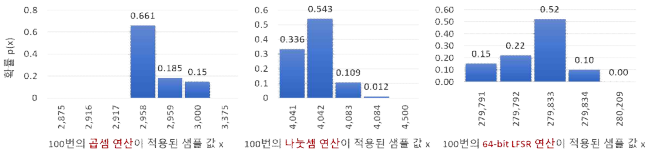


[그림 2] 시간 측정 함수에 대한 지터 엔트로피 소스의 분포

- Shannon 엔트로피 $H(X) = -\sum_x p(x)\log_2 p(x) = 1.30$-bit
- 최소 엔트로피 $H_\infty(X) = \min_x(-\log_2 p(x)) = 0.84$-bit
- NIST SP 800-90B의 최소 엔트로피 $H_\infty^{NIST}(X) = 0.19$-bit

2.2 기본 연산자가 적용된 지터 엔트로피 소스

두 번의 시간 측정 함수 사이에 100 번의 곱셈 연산(②), 나눗셈 연산(③), 64-bit LFSR 연산(④)이 각각 적용된 지터 엔트로피 소스의 분포를 도출하였고, 그 결과는 [그림 3]에 나타나 있다. 각 엔트로피 소스에 대해 세 가지의 방법으로 엔트로피를 추정하였고, 그 결과는 <표 1>에 작성되어 있다.



[그림 3] 기본 연산자에 따른 지터 엔트로피 소스의 분포

<표 1>에 작성된 엔트로피를 통해 기본 연산자는 지터 엔트로피 소스의 엔트로피에 영향을 크게 미치지 못함을 확인하였다. 또한, 기본 연산자는 샘플 수집에 대한 효율성을 감소시킴을 다음과 같은 방법으로 확인하였다. 256-bit의 엔트로피를 가진 데이터를 생성하는 경우를 가정하자. 이 경우, 추정된 엔트로피에 따라 1,348개(①), 1,829 개(②), 800 개(③) 또는 967 개(④)의 샘플을 수집해야 한다. 샘플 당 평균 수집시간을 고려하면, 1,348개의 샘플을 수집하는 데에 53 ms가 필요한 지터 엔트로피 소스 ①의 효율성이 가장 높았다. 이를 통해 Müller가 제안한 지터 엔트로피 소스 내 1 번의 64-bit LFSR 과정도 엔트로피에 영향을 미치지 않을 것이라고 분석하였다.

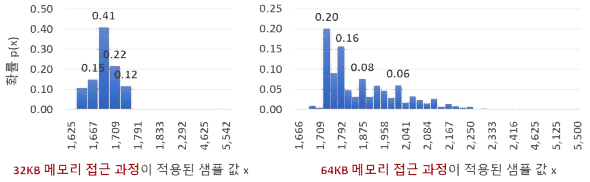
2.3 메모리 접근이 적용된 지터 엔트로피 소스

Zircon 커널은 32 KB의 L1 데이터 캐시(Data cache), 32 KB의 L1 명령어 캐시와 256 KB의 L2 캐시를 사용하고, 캐시 라인(Cache line)은 64-byte이다. L1 캐시와 L2 캐시 간 CPU의 접근 속도 차이가 있으므로, 사용하는 메모리 크기에 따른 지터 엔트로피 소스의 분포와 엔트로피를 분석하였다. 본 실험에서 사용한 메모리에 접근하는 방법은 다음과 같다. 먼저 (메모리 블록의 크기 × 블록 개수) 크기의 메모리를 할당한다. 이때, 캐시 라인을 고려하여 블록의 크기를 64-byte로 하였다. 그런 다음, (블록 크기 - 1) 단위로 움직이면서 해당 자리의 바이트 값을 1 증가시키고 모듈러(Moduler)-256 연산을 진행한다. 실험에서 사용한 메모리에 접근한 횟수는 128 번이다.

<표 1> 명령어에 따른 지터 엔트로피 소스의 최소 엔트로피 및 샘플 당 평균 수집시간

지터 엔트로피 소스		최소 엔트로피 (bit per 64-byte)	샘플 당 평균 수집시간(ns)
시간 측정 함수 ①		0.19	40
① +	100번 곱셈 ②	0.14	3,000
	100번 나눗셈 ③	0.32	4,000
	100번 64-bit LFSR ④	0.49	280,000
	32KB 메모리 128번 접근 ⑤	0.93	1,700
	64KB 메모리 128번 접근 ⑥	1.97	1,900

L1 캐시 크기인 32 KB 크기 또는 L1 캐시 크기보다 큰 64 KB 크기의 메모리에 접근하는 과정이 각각 적용된 지터 엔트로피 소스의 분포를 도출하였고, 그 결과는 [그림 4]에 나타나 있다. 각 엔트로피 소스에 대해 세 가지의 방법으로 엔트로피를 추정하였고, 그 결과는 <표 1>에 작성되어 있다.



[그림 4] 메모리 크기에 따른 지터 엔트로피 소스의 분포

<표 1>에 작성된 엔트로피를 통해 메모리에 접근하는 과정은 지터 엔트로피 소스의 엔트로피에 영향을 미칠 가능성이 있음을 확인하였다. 2.2절의 효율성 분석을 진행하였을 때, 시간 측정 함수에 대한 지터 엔트로피 소스의 효율성이 메모리 접근 과정이 적용된 경우보다 큼을 확인하였다. 본 실험은 커널 부팅 초기에 수집한 샘플에 대해 진행하였기 때문에, 사용자 인터페이스(User interface) 등의 캐시 메모리 사용이 많은 환경에서 메모리 접근 과정이 엔트로피에 미치는 영향과 수집에 대한 효율성도 커질 것으로 여겨진다. 또한, 지터 엔트로피 소스에 메모리 접근 과정을 적용할 경우 L1 캐시의 크기보다 큰 크기의 메모리를 사용함이 적절함을 확인하였다.

Ⅲ. 결론

본 논문에서는 시간 측정 함수에 대한 지터 엔트로피 소스에 대한 분포를 도출하고 엔트로피를 추정하였다. Khadas VIM2 보드에서 Zircon 커널의 부팅 초기에 수집한 샘플 데이터에 대해, 샘플 크기가 64-bit인 지터 엔트로피 소스는 0.19 비트의 엔트로피를 가짐을 확인하였다. 또한, 기본 연산자 또는 메모리 접근 과정을 적용하였을 때의 지터 엔트로피 소스에 대한 분포를 도출하고 엔트로피를 추정하였다. 그 결과, L1 캐시의 크기보다 큰 메모리를 사용하는 메모리 접근 과정이 지터 엔트로피 소스에 영향을 미칠 가능성이 큼을 확인하였다. 향후, 기본 연산자 외의 과정이 적용된 지터 엔트로피 소스를 설계하는 방법에 관해 연구를 진행할 예정이다.

ACKNOWLEDGMENT

이 성과는 2021년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No.2021M1A2A2043893)

참 고 문 헌

[1] Stephan Müller. (2020). CPU Time Jitter Based Non-Physical True Random Number Generator. https://www.chronox.de/jent/doc/CPU-Jitter-NPTRNG.html.

[2] TTA. (2020). 운영체제별 잡음원 수집 및 응용 지침. TTAK.KO-12.0235/R2.

[3] Turan, M. S., Barker, E., Kelsey, J., McKay, K., Baish, M., & Boyle, M. (2018). Recommendation for the Entropy Sources Used for Random Bit Generation. NIST Special Publication 800-90B.