Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.28, NO.4, Aug. 2018 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2018.28.4.809

윈도우 운영체제의 시간 종속 잡음원에 대한 엔트로피 평가 방법 연구*

김 예 원,[1†] 염 용 진[1,2‡]
[1]국민대학교 금융정보보안학과, [2]국민대학교 정보보안암호수학과

A Study on the Entropy Evaluation Method for Time-Dependent Noise Sources of Windows Operating System and It's Applications*

Yewon Kim,[1†] Yongjin Yeom[1,2‡]
[1]Dept. of Financial Information Security, Kookmin University,
[2]Dept. of Information Security, Cryptology and Mathematics, Kookmin University

요 약

현대 암호 시스템과 암호모듈의 필수적 요소인 난수발생기의 안전성을 평가하는 방법으로 잡음원에 대한 엔트로피 평가 방법이 있다. 국외 주요 엔트로피 평가 방법은 소프트웨어 잡음원보다 하드웨어 잡음원에 적용하는 것이 더 적합하고, 소프트웨어 잡음원별 엔트로피에 대한 정량적인 평가의 어려움이 있다. 본 논문에서는 소프트웨어 잡음원의 특징을 고려하여 소프트웨어 잡음원에 적합한 엔트로피 평가 방법을 제안하고자 한다. 윈도우 운영체제의 소프트웨어 잡음원 중에서 시간 종속 잡음원을 분석 대상으로 선정하고, 각 잡음원의 특성을 고려하여 휴리스틱(heuristic) 분석과 실험적 분석을 진행한다. 이를 기반으로 하여 시간 종속 잡음원에 대한 엔트로피 평가 방법으로 잡음원 수집 방법과 최소 엔트로피 추정 방법을 제안한다. 그리고 미국 NIST의 SP 800-90B에 기술된 Conditioning Component에 제안하는 평가 방법을 활용하는 방법을 제시한다.

ABSTRACT

The entropy evaluation method for noise sources is one of the evaluation methods for the random number generator that is the essential element of modern cryptographic systems and cryptographic modules. The primary entropy evaluation methods outside of the country are more suitable to apply to hardware noise sources than software noise sources, and there is a difficulty in quantitative evaluation of entropy by software noise source. In this paper, we propose an entropy evaluation method that is suitable for software noise sources, considering characteristics of software noise sources. We select time-dependent noise sources that are software noise sources of Windows OS, and the heuristic analysis and experimental analysis are performed considering the characteristics of each time-dependent noise source. Based on these analyses, we propose an entropy harvest method from the noise source and the min-entropy estimation method as the entropy evaluation method for time-dependent noise sources. We also show how to use our entropy evaluation method in the Conditioning Component described in SP 800-90B of NIST(USA).

Keywords: Software noise source, Entropy evaluation method, Random number generator

Received(04. 23. 2018), Modified(07. 16. 2018), Accepted(07. 18. 2018)

* 이 논문은 2018년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임(No.2014-6-00908, 난수발생기 및 임베디드 기기 안전성 연구)

† 주저자, fdt150@kookmin.ac.kr
‡ 교신저자, salt@kookmin.ac.kr(Corresponding author)

I. 서 론

난수발생기는 현대 암호 시스템과 암호모듈에서 사용하는 암호키, 난스(nonce), 씨드(seed), 보안 매개변수 등을 구성하는 필수적 요소인 난수를 생성한다. 암호 시스템과 암호모듈의 안전성을 위해 난수발생기는 예측불가능성(unpredictability), 비편향성(unbiased), 독립성(independence)이라는 특징을 가지는 암호학적으로 안전한 난수를 생성해야 하므로, 난수발생기에 대한 안전성 평가가 중요하다.

대부분의 난수발생기는 Fig. 1.에서와 같이 잡음원 수집 단계, 엔트로피 축적 및 씨드 생성 단계, 난수열 생성 단계를 통해 난수를 생성한다. 그리고 출력되는 난수가 입력인 잡음원에 의존적이라는 특징을 가진다. 이러한 특징으로 인해, 예측 가능한 잡음원 즉, 엔트로피가 낮은 잡음원을 난수발생기의 입력으로 사용하였을 때 출력되는 난수가 암호학적으로 안전한 난수의 조건을 만족하지 못한다. 그러므로 난수발생기에 대한 안전성 평가에 있어서 잡음원에 대한 평가의 비중이 크다. 가장 많이 참조되는 잡음원 평가에 대한 국외 표준문서로는 미국 NIST의 SP 800-90B[1], 독일 BSI의 AIS.31[8], ISO/IEC 20543[9] 등이 있다. 이와 같은 표준문서에서 제시하는 대부분의 난수성 평가 방법은 하드웨어 환경에서 수집 가능한 잡음원(이하 하드웨어 잡음원)과 소프트웨어 환경에서 수집 가능한 잡음원(이하 소프트웨어 잡음원)에 모두 적용 가능하다. 하지만 이 방법들은 소프트웨어 잡음원의 특징을 고려하지 않았기 때문에, 하드웨어 잡음원에 적용하는 것이 더 적합하다. 따라서 본 논문에서는 아래와 같은 소프트웨어 잡음원의 특징을 고려하여 소프트웨어 잡음원에 적합한 엔트로피 평가 방법을 제안하고자 한다.

- 소프트웨어 잡음원은 운영환경, 특히 운영체제(operating system)에 의존적임
- 소프트웨어 잡음원이 IID(Independent and Identically Distributed)이거나 균등 분포(uniform distribution)를 가지는 경우를 기대하기 어려움
- 소프트웨어 잡음원의 샘플 크기는 수 바이트에서 수백 바이트이지만, 엔트로피는 낮음
- 국외 표준문서에서 난수성 평가를 위해 요구하는 데이터 크기만큼 소프트웨어 잡음원의 샘플을 수집하기 어려움

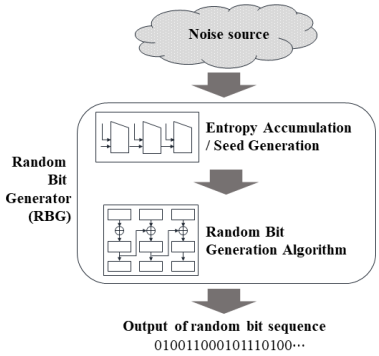

Fig. 1. The process of generating random bit sequence using RBG

- 수집하는 시간 간격에 따라 소프트웨어 잡음원의 특성이 달라질 가능성 있음

소프트웨어 잡음원에 적합한 엔트로피 평가 방법을 제안하고자, 먼저 본 논문에서는 Table 1.에 작성된 윈도우 운영체제(Windows 10(64비트))에서 수집 가능한 주요 소프트웨어 잡음원[3] 중에서 시간 종속 잡음원인 GetTickCount와 GetSystemTime을 분석 대상으로 선정하였다. 본 논문의 2장에서는 소프트웨어 잡음원의 특징을 반영하여 소프트웨어 잡음원에 대한 분석을 진행할 때 고려해야 하는 사항을 제안한다. 그리고 3장에서는 분석 대상인 각각의 잡음

Table 1. The list of the major software noise sources in Windows OS [3] (Windows 10 (64 bits))

Noise Source	Sample Size (byte)
CryptGenRandom	64
GetCurrentProcessId	4
GetCurrentThreadId	4
GetCursorPos	8
GetForegroundWindow	4
GetIcmpStatistics	104
GetIpStatistics	92
GetPerformanceInfo	56
GetProcessHeap	4
GetSystemTime	16
GetTcpStatistics	60
GetTickCount	4
GetUdpStatistics	20
GlobalMemoryStatusEx	64
HeapList	32
ProcessList	1,728
QueryPerformanceCounter	8
ThreadList	10,248

원에 대한 휴리스틱(heuristic) 분석을 진행한다. 4장에서는 실험적 분석을 진행하여 각 잡음원에 대한 엔트로피 추정 방법을 제시한다. 또한, 국외 주요 표준문서의 엔트로피 추정 방법 중 소프트웨어 잡음원에 적용하기 적합한 엔트로피 추정 방법도 제시한다. 5장에서는 휴리스틱 분석과 실험적 분석의 결과를 기반으로 하여 시간 종속 잡음원에 대한 엔트로피 평가 방법을 제안한다. 6장에서는 제안하는 엔트로피 평가 방법에 대한 활용 방안으로, 2018년 1월 10일에 발행된 NIST의 SP 800-90B[1]에서 잡음원의 엔트로피 비율(entropy rate)을 높이는 Conditioning Component를 사용하는 방법을 제시한다. 그리고 7장에서 결론을 내린다.

II. 시간 종속 잡음원에 대한 분석 과정에서 고려해야 하는 사항

본 장에서는 소프트웨어 잡음원에 대한 엔트로피 평가 방법을 제안하기 위해 분석을 진행할 때 고려해야 하는 세 가지 사항을 제안한다. 이 세 가지의 고려사항은 수집 시간 간격, 엔트로피 추정에 필요한 샘플 개수, 샘플 크기에 대한 필터로, 소프트웨어 잡음원의 특징과 엔트로피 추정 방법의 특성을 반영한 것이다.

본론에 들어가기에 앞서, 본 논문에서 사용하는 '샘플 크기'와 '샘플 개수'에 대한 용어를 정의하고자 한다. Fig. 2에서 나타난 바와 같이, 샘플 크기는 (디지털화된) 잡음원의 1회 출력으로 얻은 데이터의 크기이고, 샘플 개수는 수집 시간 간격에 따라 수집한 샘플의 개수이다.

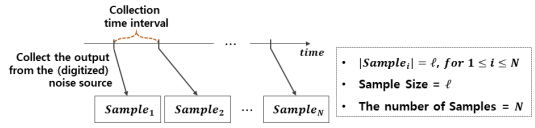

Fig. 2. Definitions of sample size and the number of samples

2.1 수집 시간 간격

본 절에서는 수집 시간 간격을 고려한 두 가지의 소프트웨어 잡음원의 수집 방법을 제안한다. 소프트웨어 잡음원은 수집 시간 간격에 따라 그 특성이 달라질 수 있으므로, 소프트웨어 잡음원을 수집/분석/평가할 때 수집 시간 간격을 필수적으로 고려해야 하기 때문이다. 제안하는 두 가지의 수집 방법은 본 논문의 3장과 4장에서의 분석 과정에 사용된다.

방법 1. 일정한 수집 간격

Fig. 3.에 있는 예시와 같이, 이 방법은 주어진 수집 시간 간격으로 일정하게 잡음원을 수집하는 것이다. Fig. 3.의 예시에서 수집 시간 간격이 항상 20 ms이므로, 평균 수집 시간 간격도 20 ms이다.

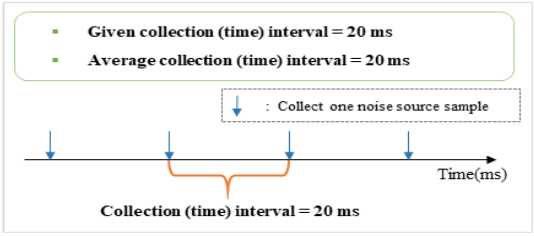

Fig. 3. Collection at regular time intervals

방법 2. 랜덤한 수집 간격

Fig. 4.에 있는 예시와 같이, 이 방법은 0 ms에서부터 주어진 최대 수집 시간 간격까지의 범위 내에서 매번 랜덤하게 선택된 시간을 수집 시간 간격으로 하여 잡음원을 수집하는 것이다. Fig. 4.의 예시에서 보이는 바와 같이, 이 경우의 수집 시간 간격의 최솟값은 0 ms이고, 최댓값은 20 ms이며, 평균 수집 시간 간격은 10 ms이다.

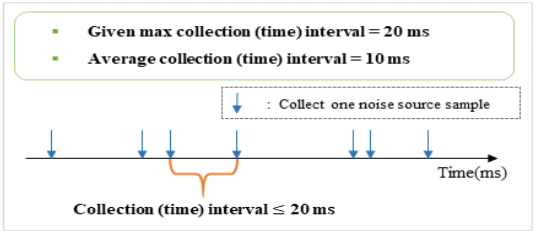

Fig. 4. Collection at random time intervals

2.2 엔트로피 추정에 필요한 샘플 개수

적은 데이터(샘플 개수)로도 소프트웨어 잡음원에 대한 의미 있는 엔트로피 추정이 가능한 방법이 필요하다. 수집 가능한 소프트웨어 잡음원의 샘플 개수는 그 잡음원의 특성과 수집 환경에 따라 다양하고, 수집 시간 간격에 영향을 받을 수 있기 때문이다. 따라서

본 논문에서는 그 방법을 모색하고자 한다. 4장에서 다양한 샘플 개수에 대한 엔트로피 추정 결과를 분석하고, 이를 이용한 평가 기준을 5장에서 제시한다.

2.3 샘플 크기에 대한 필터

크기가 1바이트를 초과하는 샘플에는 변동성이 큰 비트를 선별하고 선택하는 필터가 필요하다. 엔트로피 추정에 필요한 계산량과 메모리량은 잡음원으로부터 출력 가능한 모든 샘플의 개수($=2^n$, n: 샘플의 비트 크기)에 영향을 받으므로, 주요 표준문서에서 제시하는 대부분의 엔트로피 추정 방법[1, 2]이 크기가 1바이트 이하인 샘플에 대해서만 적용 가능하기 때문이다. SP 800-90B의 6.4절에서는 일반적인 필터 사용 방법을 설명하고, 수학적인 함수 등으로 샘플을 가공하지 않는 방식을 적용하는 것을 요구한다[1].

본 절에서는 바이트 필터를 제안하고, 3장과 4장에서 이를 크기가 수 바이트에서 수백 바이트인 소프트웨어 잡음원에 적용하여 분석을 진행한다. 제안하는 바이트 필터는 수집되는 샘플에서 가장 값이 많이 변하는 최적의 바이트 위치를 선별하고 그 위치에 있는 값을 추출하여 1바이트 샘플로 변환하는 과정이다.

III. 시간 종속 잡음원에 대한 휴리스틱 분석

본 장에서는 윈도우 운영체제의 시간 종속 잡음원인 GetTickCount와 GetSystemTime에 대하여 MSDN[5]에 기반을 둔 구조 분석을 통해 휴리스틱 분석을 진행한다. 이때, 미국 NIST의 암호모듈 검증제도에 대한 구현가이드(이하 CMVP IG)[4]에 기술된 시간 관련 잡음원에 대한 휴리스틱 분석 방법을 보수적으로 사용하고, [3, 6]에서 진행한 휴리스틱 분석 방법을 활용한다.

CMVP IG에 기술된 시간 관련 잡음원에 대한 휴리스틱 분석 방법[4]은 다음과 같다.

시, 분, 초, 밀리세컨드(milliseconds, 이하 ms)로 구성된 시간 관련 잡음원을 hh:mm:ss:zzz로 표현하였을 때, 잡음원의 구성 요소 중 가장 많이 변하는 값은 ms인 zzz($0 \le zzz \le 999$, $0 \le z \le 9$)이다. 따라서 zzz 값의 변동성을 통해 시간 관련 잡음원의 엔트로피를 추정하고, 측정 빈도에 따라 추정 엔트로피가 달라짐을 고려한다. 매번 서로 다른 빈도로 잡음원을 수집하는 경우에는 zzz가 1,000개의 서로 다른 값을 가질 수 있으므로, 시간 관련 잡음원의 엔트로피를 약 10비트로 추정한다. 하지만 현실적으로 매번 서로 다른 빈도로 수집하는 것은 어렵다. 따라서 이를 고려하여 매번 0.5초 정도의 간격으로 수집하는 경우, 변동 가능성이 있는 두 번째 z와 세 번째 z로 인해 zzz는 약 100개의 서로 다른 값을 가질 수 있다. 또한, 첫 번째 z의 변동 가능성도 고려하여 약 128개의 서로 다른 값을 zzz가 가질 수 있다고 여긴다. 이로 인해, 이 경우의 시간 관련 잡음원의 엔트로피를 약 7비트로 추정한다.

3.1 GetTickCount 구조 및 휴리스틱 분석

3.1.1 GetTickCount 구조

GetTickCount는 운영체제가 시작된 이후 경과한 시간으로, 측정 단위는 ms이다. GetTickCount의 샘플 크기는 4바이트이고, GetTickCount() 함수가 수집 함수이다. GetTickCount() 함수의 정밀도는 10 ms ~ 16 ms로, 이는 10 ms ~ 16 ms인 범위 내에 있는 ms만큼의 시간이 지나서야 한 번씩 업데이트된다는 것을 의미한다.

3.1.2 GetTickCount에 대한 휴리스틱 분석

수집 함수의 정밀도에 의해 GetTickCount는 10 ms ~ 16 ms인 범위 내에 있는 시간만큼씩 업데이트되므로, 4바이트의 샘플 크기인 GetTickCount에서 가장 값이 많이 변하는 최적의 바이트 위치는 최하위 바이트 자리이다. 따라서 본 절에서는 CMVP IG와 유사하게 최하위 바이트 값의 변동성을 이용하여 GetTickCount에 대한 엔트로피를 추정한다. 이때, 2장에서 제안한 두 가지의 수집 방법을 적용하여 진행한다. 또한, 정밀도가 특정 값으로 고정되어 있다는 가정을 하고, 가정에 따라 분석을 진행한다. 이는 수집 환경에 영향을 받는 수집 함수의 정밀도에 따라 휴리스틱 분석으로 추정되는 GetTickCount의 엔트로피가 달라지기 때문이다. 본 절에서는 정밀도에 대한 두 가지의 가정을 하고, GetTickCount() 함수의 정밀도를 R이라는 변수로 정의한다. 첫 번째 가정은 R = 15.625 ms로, 윈도우 운영체제의 기본 타이머의 정밀도와 동일하다는 것이다. 두 번째 가정은 R = 10 ms로, GetTickCount() 함수의 가장 높은 정밀도로 고정되어있다는 것이다.

1. R = 15.625 ms 가정 하에서의 엔트로피 추정

1) '일정한 수집 간격' 방법으로 수집하는 경우

주어진 수집 시간 간격으로 일정하게 수집하면 GetTickCount 샘플의 최하위 바이트 값은 $15.625 \times \left\lfloor \frac{\text{일정한 수집 시간 간격}}{15.625} \right\rfloor$ 만큼 일정하게 업데이트되므로, 최하위 바이트 값의 변동성은 1 개이다. 따라서 GetTickCount의 엔트로피를 0 비트로 추정한다.

2) '랜덤한 수집 간격' 방법으로 수집하는 경우

주어진 수집 시간 범위 내에서 매번 랜덤하게 선택된 시간 간격으로 수집하고 CMVP IG의 휴리스틱 분석 방법을 보수적으로 활용하면, GetTickCount 샘플의 최하위 바이트는 $\left\lfloor \frac{\text{최대 수집 시간 간격}}{15.625} \right\rfloor$ 개의 서로 다른 값을 가질 수 있다. 따라서 GetTickCount의 엔트로피를 Table 2.와 같이 추정한다. 그리고 엔트로피를 추정한 방법에 대한 예시는 다음과 같다.

최대 수집 시간이 100 ms일 경우, 0 ms ~ 100 ms인 범위 내에서 매번 랜덤하게 선택된 시간 간격으로 GetTickCount가 수집된다. 수집 함수의 정밀도(=R)가 15.625 ms로 고정되어있다는 가정으로 인해, GetTickCount는 수집되는 시간 간격에 따라 15.625 ms, 31.25 ms, 46.875 ms, 62.5 ms, 78.125 ms, 93.75 ms 에서 업데이트된 시간이 반영된다. 따라서 GetTickCount의 최하위 바이트가 가질 수 있는 서로 다른 값의 개수는 $\left\lfloor \frac{100(=\text{최대 수집 시간 간격})}{15.625} \right\rfloor = 6$ 개다. 그리고 보수적으로 GetTickCount의 엔트로피를 2 비트로 추정한다.

Table 2. The estimated entropy of GetTickCount when collecting data at 'random time intervals' method (Assumption: R = 15.625 ms)

Given collection time interval(t)	Estimated entropy
$1 \le t < 31.25$	0 bit of entropy
$31.25 \le t < 62.5$	1 bit of entropy
$62.5 \le t < 125$	2 bits of entropy
$125 \le t < 250$	3 bits of entropy
$250 \le t$	4 bits of entropy

2. R = 10 ms 가정 하에서의 엔트로피 추정[3]

1) '일정한 수집 간격' 방법으로 수집하는 경우

앞서 살펴본 바와 동일하게 GetTickCount의 엔트로피를 추정하면 0 비트이다.

2) '랜덤한 수집 간격' 방법으로 수집하는 경우

R = 10 ms인 가정에 따라 앞서 살펴본 바와 동일하게 엔트로피를 추정한 결과는 Table 3.과 같다.

Table 3. The estimated entropy of GetTickCount when collecting data at 'random time intervals' method (Assumption: R = 10 ms)

Given collection time interval(t)	Estimated entropy
$1 \le t < 20$	0 bit of entropy
$20 \le t < 40$	1 bit of entropy
$40 \le t < 80$	2 bits of entropy
$80 \le t < 160$	3 bits of entropy
$160 \le t$	4 bits of entropy

Table 4. The components of GetSystemTime [5] and the byte position

Component	Description	Byte position
wYear	The year (1601 ~ 30827)	S[0], S[1]
wMonth	The month (1 ~ 12)	S[2], S[3]
wDayOfWeek	The day of week (0 ~ 6)	S[4], S[5]
wDay	The day (1 ~ 31)	S[6], S[7]
wHour	The hour (0 ~ 13)	S[8], S[9]
wMinute	The minute (0 ~ 59)	S[10], S[11]
wSecond	The second (0 ~ 59)	S[12], S[13]
wMilliseconds	The millisecond (0 ~ 999)	S[14], S[15]

3.2 GetSystemTime 구조 및 휴리스틱 분석

3.2.1 GetSystemTime 구조

GetSystemTime은 UTC+0인 현재 시스템의 날짜와 시간이다. GetSystemTime의 샘플 크기는

16 바이트이고, 수집 함수는 GetSystemTime() 함수이다. GetSystemTime는 8 개의 크기가 2 바이트인 요소로 구성되어 있고, 각 구성 요소는 리틀 엔디안(little endian) 방식으로 저장된다. GetSystemTime을 크기가 16인 바이트 열 S로 보았을 때, Table 4.는 GetSystemTime의 각각의 구성 요소와 그 값이 저장되는 바이트 자리를 설명한다.

3.2.2 GetSystemTime에 대한 휴리스틱 분석

GetSystemTime에서 가장 값이 많이 변하는 구성 요소는 ms인 wMilliseconds이다. 그리고 크기가 2 바이트인 wMilliseconds에서 가장 값이 많이 변하는 최적의 바이트 위치는 최하위 바이트 자리(S[14])이다. 따라서 CMVP IG와 유사하게 wMilliseconds의 최하위 바이트 값의 변동성을 이용하여 GetSystemTime에 대한 엔트로피를 추정한다. 이때, 2 장에서 제안한 두 가지의 수집 방법을 적용하여 진행한다.

1. '일정한 수집 간격' 방법으로 수집하는 경우

주어진 수집 시간 간격으로 일정하게 수집하면 wMilliseconds의 최하위 바이트 값은 수집 시간 간격만큼 일정하게 업데이트된다. 최하위 바이트 값의 변동성은 1 개이므로, GetSystemTime의 엔트로피를 0 비트로 추정한다.

2. '랜덤한 수집 간격' 방법으로 수집하는 경우

주어진 수집 시간 범위 내에서 매번 랜덤하게 선택된 시간 간격으로 수집하고 CMVP IG의 휴리스틱 분석 방법을 보수적으로 활용하면, wMilliseconds의 최하위 바이트는 0 개 또는 16 개의 서로 다른 값을 가질 수 있다. 따라서 GetSystemTime의 엔트로피를 Table 5.와 같이 추정한다. 그리고 엔트로피를 추정한 방법에 대한 예시는 다음과 같다.

먼저 CMVP IG의 휴리스틱 분석 과정에서 ms를 10 진수인 zzz로 표현한 바와 같이, wMilliseconds의 최하위 바이트를 16 진수인 xx로 표현한다.

주어진 수집 시간 간격이 30 ms인 경우, 0 ms ~ 30 ms인 범위 내에서 매번 랜덤하게 선택된 시간 간격으로 GetSystemTime이 수집되므로, 변동 가능성이 있는 두 번째 x로 인해 xx는 약 16 개의 서로 다른 값을 가질 수 있다. 그리고 CMVP IG보다 보수적으로 엔트로피를 추정하고자 첫 번째 x의 변동성을 고려하지 않는다. 따라서 GetSystemTime의 엔트로피를 4 비트로 추정한다.

주어진 수집 시간 간격이 10 ms인 경우, 0 ms ~ 10 ms인 범위 내에서 매번 랜덤하게 선택된 시간 간격으로 수집된다. 이 경우, GetSystemTime에서 증가하는 값이 크지 않기 때문에, 카운터의 특성을 가질 수 있다. 그러므로 엔트로피를 더 보수적으로 추정하고자 첫 번째와 두 번째 x의 변동성을 고려하지 않고, GetSystemTime의 엔트로피를 0 비트로 추정한다.

Table 5. The estimated entropy of GetSystemTime when collecting data at 'random time intervals' method[3]

Given collection time interval(t)	Estimated entropy
$1 \le t < 16$	0 bit of entropy
$16 \le t$	4 bits of entropy

IV. 시간 종속 잡음원에 대한 실험적 분석

본 장에서는 GetTickCount와 GetSystemTime을 수집 옵션에 따라 수집하고, 국외 주요 표준문서의 엔트로피 추정 알고리즘 중에서 소프트웨어 잡음원에 적용하기 적합하다고 여겨지는 알고리즘으로 각 잡음원의 실제 최소 엔트로피를 추정한다. 이를 통해 각 잡음원에 대한 엔트로피 추정 방법을 제시한다.

4.1 실험 환경

- 운영체제 : Windows 10 Pro x64
- 프로세서 : Intel(R) Core(TM) i7-4790K CPU
- RAM : 32.0 GB
- GetTickCount() 함수의 정밀도 : 15.625 ms

4.2 실험 방법

4.2.1 수집 옵션

Table 6.과 같이 수집 시간 간격과 수집할 샘플 개수를 설정하여 잡음원을 수집한다. 이때, 2 장에서 제안한 '일정한 수집 간격' 방법과 '랜덤한 수집 간격' 방법을 각각 적용한다.

Table 6. Collection time interval and the number of samples to collect

Collection time interval (ms)	10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200, 210, 220, 230, 240, 250, 260, 270, 280, 290, 300, 400, 500
The number of samples to collect	1,000, 2,000, 3,000, 4,000, 5,000, 6,000, 7,000, 8,000, 9,000, 10,000, 20,000, 50,000

4.2.2 최소 엔트로피 추정 알고리즘

소프트웨어 잡음원인 시간 종속 잡음원에 대한 최소 엔트로피를 추정하기 위해, 국외 주요 표준문서인 SP 800-90B[1, 10]의 최소 엔트로피 추정 알고리즘 중에서 적은 입력 데이터로도 Non-IID 잡음원에 대한 엔트로피 추정이 가능하다고 여겨지는 최빈값(the most common value) 추정, 충돌(collision) 추정, 압축(compression) 추정 알고리즘을 선정하였다. 이때, 최종판인 SP 800-90B[1]의 충돌 추정과 압축 추정 알고리즘은 크기가 1비트인 잡음원에 대해서만 적용 가능하므로, 크기가 8비트인 잡음원에 대해서도 적용 가능한 SP 800-90B(2nd Draft)[10]의 충돌 추정과 압축 추정 알고리즘을 사용하였다.

본 실험에서는 앞서 선정한 세 가지의 최소 엔트로피 추정 알고리즘의 결과 중에서 최솟값을 입력인 잡음원의 최소 엔트로피로 추정한다[3, 6]. 이때, 충돌 추정과 압축 추정 알고리즘은 수행 과정에서 수치해석(numerical analysis)에 따라 비선형함수의 해를 탐색하는 것을 실패하면, 엔트로피의 저하를 발견하지 못한 것으로 판단하여 잡음원이 풀 엔트로피(full entropy)를 가진다고 판정한다. 이러한 결과는 비정상적인 엔트로피 추정으로 간주할 수 있으므로, 이 경우에는 다른 엔트로피 추정 알고리즘을 사용하는 것이 적절하다. 따라서 본 실험에서는 풀 엔트로피를 출력하지 않은 엔트로피 추정 알고리즘의 결과 중에서 최솟값을 계산하여 최소 엔트로피를 추정한다[3]. 그리고 결과를 나타내는 막대 그래프인 Fig. 6. ~ 11., Fig. 14., Fig. 17.에서 풀 엔트로피를 출력하는 알고리즘을 제외하여 추정한 최소 엔트로피의 막대를 ▭ 굵게 표현하였다.

4.2.3 실험 시나리오

1. 바이트 필터 위치 선정

수집 옵션을 적용하여 수집한 데이터에 바이트 필터 계산 방법을 사용하여 최적의 바이트 위치를 선별한다. 바이트 필터 계산 방법[3, 6]으로, 먼저 1,024개의 잡음원 샘플의 각 바이트 열에 AIS.31[8]의 모노비트(monobit) 테스트와 포커-8(poker-8) 테스트를 사용한다. 그런 다음, 각 바이트 열의 테스트 결과로 순위를 선정하여 가장 순위가 높은 바이트 열, 다시 말해, 난수성이 가장 높은 바이트 열을 선택한다. 만약 동점자가 발생하면 하위 바이트 열을 선택한다. 이 방법을 통해 3장에서 제시한 각 잡음원의 최적의 바이트 위치에 대한 타당성을 검증한다.

2. 동일한 실험 조건 하에서 반복 실험 결과의 변동폭 비교

동일한 수집 옵션을 적용하고 동일한 최적의 바이트 위치의 값으로 추출한 잡음원의 최소 엔트로피를 추정하는 실험을 5번 반복 진행한다. 각 실험 결과의 변동폭을 비교하는 과정을 통해 본 절에서 진행한 실험 방법 및 결과와 제안하는 엔트로피 추정 방법의 타당성을 검증한다.

3. 실험 결과와 휴리스틱 분석 결과 비교

실제 수집한 잡음원의 최소 엔트로피를 추정한 실험 결과와 휴리스틱 분석으로 추정한 엔트로피를 비교한다. 이를 통해 선정한 최소 엔트로피 추정 알고리즘에 대한 타당성을 검증하고, 각 잡음원에 적합한 엔트로피 추정 방법을 제안한다.

4.3 GetTickCount에 대한 실험 및 타당성 검증

4.3.1 바이트 필터 위치 선정

크기가 4바이트인 GetTickCount 샘플을 크기가 4인 바이트 열 S로 구성한다. 만약 GetTickCount 샘플의 값이 0x0105FA87인 경우라면, 바이트 열 S를 {0x01, 0x05, 0xFA, 0x87}로 구성한다.

2장에서 제안한 두 가지의 수집 방법에 Table 6.에 있는 32개의 수집 시간 간격을 각각 적용하여 수집한 GetTickCount에 바이트 필터 계산 방법을 사용하여 최적의 바이트 위치를 계산하는 과정을 4

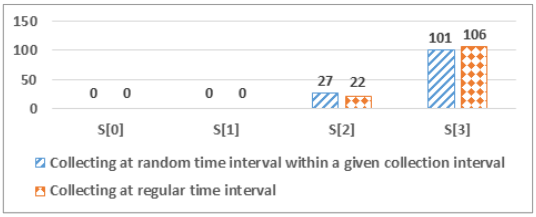

Fig. 5. The results of the byte filter calculation on GetTickCount (X-axis: byte position, Y-axis: the number of results calculated with the byte position)

번 반복 진행하였다.

Fig. 5.에서 보이는 바와 같이, 수집 시간 간격과 수집 방법과는 상관없이 바이트 필터 계산을 통해 얻은 GetTickCount의 최적의 바이트 위치는 대부분 최하위 바이트 자리인 S[3] 자리이다. S[2] 자리가 최적의 바이트 위치로 계산될 가능성은 수집 시간 간격의 크기가 커질수록 높아진다. 하지만, 수집 시간 간격의 크기와 상관없이, S[3] 자리의 값을 추출하여 추정한 최소 엔트로피가 S[2] 자리일 때보다 높음을 Table 7.을 통해 알 수 있다.

Table 7.은 최대 수집 시간이 120 ms 또는 270 ms로 주어진 시간 범위 내에서 랜덤하게 선택된 시간 간격으로 7,000개의 샘플을 수집하고, S[2] 자리와 S[3] 자리의 값을 각각 추출하여 추정한 최소 엔트로피를 보여준다. 따라서 GetTickCount의 최적의 바이트 위치는 S[3]인 최하위 바이트 자리이고, 이는 수집 방법과 수집 시간 간격에 영향을 받지 않는다. 또한, 3장에서의 휴리스틱 분석이 GetTickCount에 적합하게 진행되었음을 확인할 수 있다.

Table 7. The estimated min-entropy by applying byte filter position S[2] or S[3] (Given max collection time interval = 120 or 270)

byte filter position / max collection time interval	S[2]	S[3]
120 ms	0.18	3.45
270 ms	0.47	5.87

4.3.2 동일한 실험 조건 하에서 반복 실험 결과의 변동폭 비교

2장에서 제안한 두 가지 수집 방법에 Table 6.에 있는 수집 옵션을 적용하여 GetTickCount를 수집하였고, 각 옵션에 따라 수집한 GetTickCount에서 최적의 바이트 위치인 최하위 바이트 자리의 값을 추출하여 최소 엔트로피를 추정하는 실험을 5번 반복 진행하였다. 세 가지의 알고리즘 모두 풀 엔트로피를 출력하지 않았을 때의 실험 결과만 분석하여 각 옵션에 대한 5번 반복 실험 결과의 평균 변동률이 10.8 %임을 확인하였다. 이는 동일한 수집 옵션을 적용하여 GetTickCount를 수집하고 최하위 바이트 값을 추출하여 최소 엔트로피를 추정할 때마다 그 결과인 최소 엔트로피가 큰 변동 없이 유사함을 의미한다. 또한, 이는 GetTickCount에 대한 실험 방법 및 결과와 제안하는 엔트로피 추정 방법이 타당함을 보여준다.

Fig. 6.과 Fig. 7.은 수집 방법에 따라 Table 8.에 있는 수집 시간 간격과 샘플 개수를 적용하여 GetTickCount 샘플을 수집하고 최하위 바이트 값을 추출하여 최소 엔트로피를 추정하는 실험을 5번 반복 진행한 결과이다. 그 결과, 동일한 수집 옵션을 적용하여 수집하고 최소 엔트로피를 추정할 때마다 평균 변동률이 7.6 %인 GetTickCount의 최소 엔트로피를 추정할 수 있다. Table 9.는 Fig. 6.에서의 수집 시간 간격에 따른 5번 반복 실험 결과의 평균과 변동폭, 변동률을 나타낸다. 이때, 해 탐색이 실패하여 풀 엔트로피를 출력하는 알고리즘이 있는 실험의 최소 엔트로피는 제외하였다.

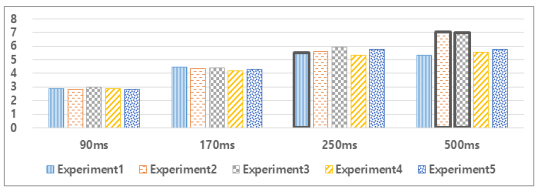

Fig. 6. The comparison of five entropy estimations of GetTickCount when collecting data at 'random time intervals' method and using the same collection options (X-axis: given max collection time interval, Y-axis: min-entropy)

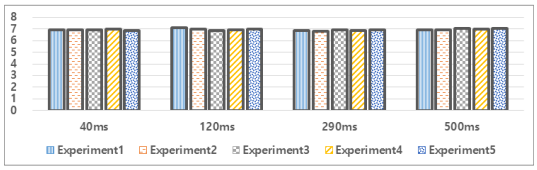

Fig. 7. The comparison of five entropy estimations of GetTickCount when collecting data at 'regular time intervals' method and using the same collection options (X-axis: given regular collection time interval, Y-axis: min-entropy)

Table 8. Collection time interval and the number of samples according to the collection method

	Collection time interval (ms)	The number of samples
Random time intervals method (Fig. 6.)	90, 170, 250, 500	9,000
Regular time intervals method (Fig. 7.)	40, 120, 290, 500	4,000

Table 9. Mean, variation and variation ratio of five entropy estimations according to collection interval in Fig. 6.

Collection interval	Average	Variation (max)	Variation ratio (%)
90 ms	2.89	± 0.1	6.04
170 ms	4.33	± 0.14	5.98
250 ms	5.66	± 0.33	10.31
500 ms	5.55	± 0.22	8.07

4.3.3 실험 결과와 휴리스틱 분석 결과 비교

앞서 동일한 수집 옵션을 적용하여 최소 엔트로피를 추정할 때마다 유사한 최소 엔트로피로 추정됨을 확인하였다. 따라서 본 절에서는 한 번의 실험으로 추정한 GetTickCount의 최소 엔트로피와 3 장에서 휴리스틱 분석으로 추정한 GetTickCount의 엔트로피를 비교하는 과정을 수집 방법에 따라 진행한다. 이를 통해 선정한 최소 엔트로피 추정 알고리즘에 대한 타당성을 검증하고, GetTickCount의 엔트로피 추정 방법을 제시한다.

1. '랜덤한 수집 간격' 방법으로 수집한 경우

Table 6.에 있는 수집 시간 간격과 샘플 개수를 적용하여 주어진 시간 범위 내에서 랜덤하게 선택된 시간 간격으로 GetTickCount를 수집하였다. 그리고 각 옵션에 따라 수집한 데이터에서 샘플의 최하위 바이트 값을 추출하고, 최소 엔트로피 추정 알고리즘으로 최소 엔트로피를 추정하였다. Fig. 8.은 수집 옵션을 다음과 같이 적용하였을 때 추정한 최소 엔트로피를 나타낸다. 이때, 실험을 진행한 환경에서 수집 함수의 정밀도(=R*)는 15.625 ms이다.

- 수집 시간 간격(ms)
  : 10, 30, 50, 100, 160, 260, 400
- 수집할 샘플 개수
  : 1,000, 2,000, 7,000, 10,000, 50,000

Fig. 8.에서 보이는 바와 같이, 최대 수집 시간이 클수록 GetTickCount의 추정된 최소 엔트로피가 증가한다. 또한, 적어도 2,000 개의 샘플을 수집하여 최하위 바이트 값을 추출한 데이터를 최소 엔트로피 추정 알고리즘에 사용하면, 의미 있는 최소 엔트로피가 추정된다. 그리고 실험을 통해, 최대 수집 시간과 수집 샘플 개수가 증가할수록 최소 엔트로피 추정 알고리즘 중에서 적어도 하나의 알고리즘(충돌 추정 또는 압축 추정)이 해 탐색을 실패하여 풀 엔트로피로 추정할 가능성이 높아짐을 알 수 있다.

R = 15.625 ms이거나 R = 10 ms인 각 가정에 따라 휴리스틱 분석으로 추정한 엔트로피와 실험으로 추정한 엔트로피를 비교하였고, 그 결과를 Fig. 9.와 Fig. 10.에 제시하였다.

Fig. 9.에서와 같이, R = 15.625 ms인 가정에 따라 보수적으로 진행한 휴리스틱 분석의 결과(Table 2.)인 엔트로피는 R* = 15.625 ms인 실제 환경에서 진행한 실험 결과인 최소 엔트로피보다 낮으나, 실험 결과가 휴리스틱 분석 결과를 잘 따르고 있음을 알 수 있다. 또한, 이를 통해 선정한 추정 알고리즘이 타당

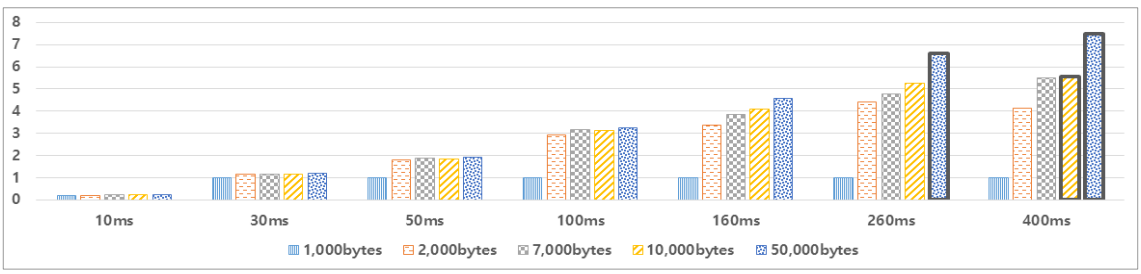

Fig. 8. The estimated min-entropy of GetTickCount when collecting data at 'random time intervals' method (R* = 15.625 ms, X-axis: given max collection time interval, Y-axis: min-entropy)

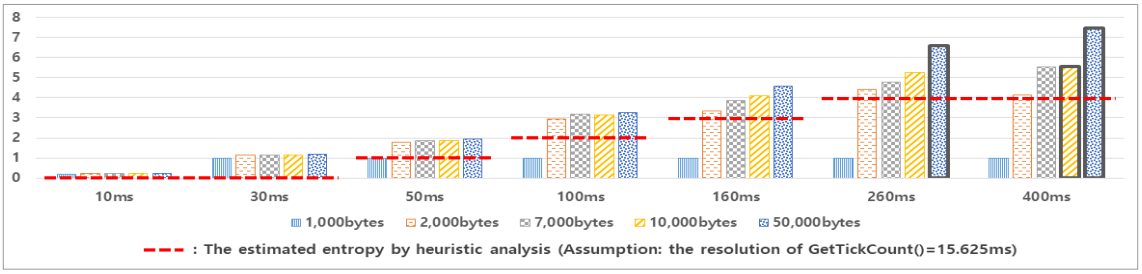

Fig. 9. The comparison of the estimated min-entropy by experiment and the estimated entropy by heuristic analysis when collecting data at 'random time intervals' method (R^* = 15.625 ms, Assumption: R = 15.625 ms, X-axis: given max collection time interval, Y-axis: min-entropy)

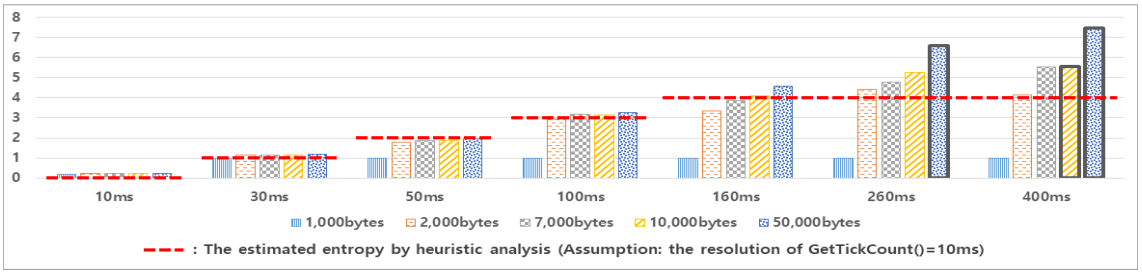

Fig. 10. The comparison of the estimated min-entropy by experiment and the estimated entropy by heuristic analysis when collecting data at 'random time intervals' method (R^* = 15.625 ms, Assumption: R = 10 ms, X-axis: given max collection time interval, Y-axis: min-entropy)

함을 알 수 있다. 이때, 샘플 개수가 1,000 개인 실험 결과는 의미가 없다고 판단하여 분석에서 제외한다.

반면, Fig. 10.에서와 같이, 대부분의 실험 결과가 R = 10 ms인 가정에 따라 진행한 휴리스틱 분석의 결과(Table 3.)를 잘 따르고 있으나, 실험 결과보다 휴리스틱 분석으로 추정한 엔트로피가 높을 수 있음을 알 수 있다. 이는 실험 환경에서의 정밀도(=R^*)보다 높은 정밀도를 가정하고 휴리스틱 분석을 진행하였기 때문에 실제 잡음원이 가지고 있는 엔트로피보다 과추정한 결과로 볼 수 있다. 따라서 휴리스틱 분석을 진행할 때 실험 환경에 대한 정확한 분석이 필요하다.

Fig. 9.와 Fig. 10.에서의 결과를 통해 다음과 같은 GetTickCount에 대한 엔트로피 추정 방법을 제안한다.

- 적어도 2,000 개의 샘플을 수집하여 엔트로피 추정 알고리즘의 입력으로 사용한다.
- 휴리스틱 분석으로 추정한 엔트로피와 실험으로 추정한 최소 엔트로피 중에서 최솟값을 랜덤한 시간 간격으로 수집한 GetTickCount의 최소 엔트로피로 추정한다[3].

$$Min\,Entropy = \min\{Entropy_{Heuristic}, Entropy_{\mathrm{Exp}}\}$$

이는 동일한 수집 옵션을 적용하여 수집할 때마다 추정한 최소 엔트로피의 변동률과 예기치 않은 오류가 있는 실험 환경에서 수집한 잡음원이 가지는 최소 엔트로피를 고려한 것이다.

2. '일정한 수집 간격' 방법으로 수집한 경우

Table 6.에 있는 수집 옵션을 적용하여 각각의 수집 시간 간격으로 일정하게 GetTickCount를 수집하였다. 그리고 각 옵션에 따라 수집한 데이터에서 샘플의 최하위 바이트 값을 추출하여 최소 엔트로피를 추정하였다.

3 장에서 진행한 일정한 수집 시간 간격일 때의 휴리스틱 분석 결과를 보면, 정밀도에 대한 가정과는 상관없이 추정 엔트로피는 항상 0 비트이다. Fig. 11.은 휴리스틱 분석으로 추정한 엔트로피와 실험으로 추정한 최소 엔트로피를 비교한 결과로, 수집 옵션은 다음과 같이 적용하였다.

- 수집 시간 간격(ms) : 40, 100, 250, 500
- 수집할 샘플 개수 : 3,000, 9,000, 20,000

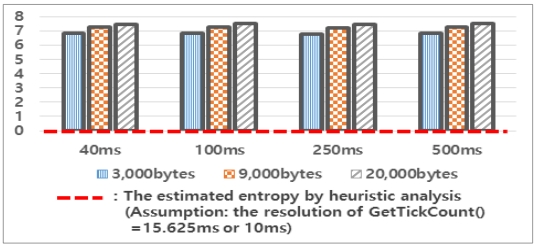

Fig. 11. The comparison of the estimated min-entropy by experiment and the estimated entropy by heuristic analysis when collecting data at 'regular time intervals' method (R* = 15.625 ms, Assumption: R = 15.625 ms or 10 ms, X-axis: given max collection time interval, Y-axis: min-entropy)

Fig. 11.에서 보이는 바와 같이, 대부분의 수집 옵션에 대해 충돌 추정과 압축 추정 알고리즘이 해 탐색을 실패하여 풀 엔트로피로 추정한다. 또한, 일정한 수집 시간 간격으로 수집하였기 때문에 이 경우의 GetTickCount는 연속적으로 일정한 값이 증가하는 카운터 특징을 가지게 된다. 따라서 충돌 추정과 압축 추정 알고리즘이 모두 해 탐색을 실패하는 경우, 실험으로 추정한 엔트로피가 휴리스틱 분석으로 추정한 엔트로피인 0 비트와 같다고 여기는 것이 적절하다.

이를 통해 제시하는 GetTickCount의 엔트로피 추정 방법으로는 휴리스틱 분석으로 추정한 엔트로피와 실험으로 추정한 최소 엔트로피 중에서 최솟값을 GetTickCount의 최소 엔트로피로 추정하는 것[3]이다. 따라서 주어진 일정한 시간 간격으로 수집한 경우, GetTickCount의 최소 엔트로피는 항상 0 비트이다.

$$Min\,Entropy = \min\{Entropy_{Heuristic}, Entropy_{\mathrm{Exp}}\} = 0\,bit$$

4.4 GetSystemTime에 대한 실험 및 타당성 검증

4.4.1 바이트 필터 위치 선정

GetTickCount와 동일하게, 크기가 16 바이트인 GetSystemTime 샘플을 크기가 16인 바이트 열 S로 구성한다. 또한, 두 가지의 수집 방법과 Table 6.에 있는 32 개의 수집 시간 간격을 적용하여 최적의 바이트 위치를 계산하는 과정을 4 번 반복 진행하였다.

Fig. 12.의 그래프에서 바이트 필터 계산 결과로 나오지 않은 바이트 자리는 생략하였다. Fig. 12.에

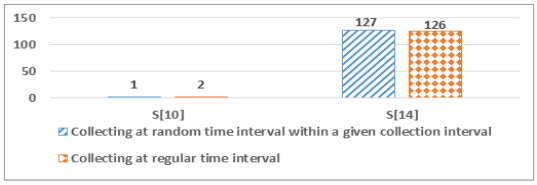

Fig. 12. The results of the byte filter calculation on GetSystemTime (X-axis: byte position, Y-axis: the number of results calculated with the byte position)

서 보이는 바와 같이, 수집 시간 간격과 수집 방법과는 상관없이 바이트 필터 계산을 통해 얻은 GetSystemTime의 최적의 바이트 위치는 S[14] 자리이다. Table 4.에서와 같이 GetSystemTime의 구성 요소 중 ms인 wMilliseconds는 S[14]와 S[15] 자리에 리틀 엔디안 방식으로 저장되기 때문에, S[14]에는 wMilliseconds의 최하위 바이트 값이 저장된다. 따라서 3 장에서 진행한 휴리스틱 분석이 GetSystemTime에 적합하게 진행되었음을 확인할 수 있다.

4.4.2 동일한 실험 조건 하에서 반복 실험 결과의 변동폭 비교 바이트 필터 자리 분석

GetTickCount에서와 동일하게, 각 옵션에 따라 수집한 GetSystemTime에서 최적의 바이트 위치인 S[14] 자리의 값을 추출하여 최소 엔트로피를 추정하는 실험을 5 번 반복 진행하였다. 세 가지의 알고리즘이 모두 풀 엔트로피를 출력하지 않았을 때의 실험 결과만 분석하여 각 옵션에 대한 5 번 반복 실험 결과의 평균 변동률이 11.34 %임을 확인하였다. 이는 동일한 옵션을 적용하여 최소 엔트로피를 추정할 때마다 그 결과인 최소 엔트로피가 큰 변동 없이 유사함을 의미한다. 또한, GetSystemTime에 대한 실험 방법 및 결과와 제안하는 엔트로피 추정 방법이 타당함을 보여준다.

Fig. 13.과 Fig. 14.는 수집 방법에 따라 Table 10.에 있는 수집 시간 간격과 샘플 개수를 적용하여 샘플을 수집하고 S[14] 자리의 값을 추출하여 최소 엔트로피를 추정하는 실험을 5 번 반복 진행한 결과이다. Fig. 13.과 Fig. 14.에서 보이는 바와 같이, 동일한 수집 옵션을 적용하여 수집하고 최소 엔트로피를 추정할 때마다 평균 변동률이 10.14 %인 GetSystemTime의 최소 엔트로피를 추정할 수 있다. Table 11.은 Fig. 13.에서의 수집 시간 간격에 따른 5 번 반복 실험 결과의 평균과 변동폭, 변동률을 나타낸다. 이때, 해 탐색이

실패하여 풀 엔트로피를 추정하는 알고리즘이 있는 실험의 최소 엔트로피는 제외하였다.

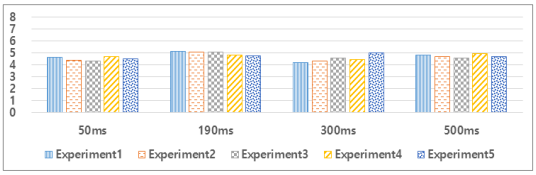

Fig. 13. The comparison of five entropy estimations of GetSystemTime when collecting data at 'random time intervals' method and using the same collection options (X-axis: given max collection time interval, Y-axis: min-entropy)

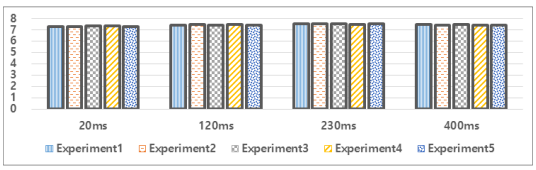

Fig. 14. The comparison of five entropy estimations of GetSystemTime when collecting data at 'regular time intervals' method and using the same collection options (X-axis: given regular collection time interval, Y-axis: min-entropy)

Table 10. Collection time interval and the number of samples according to the collection method

	Collection time interval (ms)	The number of samples
Random time intervals method (Fig. 13.)	50, 190, 300, 500	8,000
Regular time intervals method (Fig. 14.)	20, 120, 230, 240	20,000

Table 11. Mean, variation and variation ratio of five entropy estimations according to collection interval in Fig. 13.

Collection interval	Average	Variation (max)	Variation ratio (%)
50 ms	4.51	± 0.18	8.3
190 ms	4.98	± 0.19	6.61
300 ms	4.53	± 0.48	17.51
500 ms	4.76	± 0.23	8.16

4.4.3 실험 결과와 휴리스틱 분석 결과 비교

앞서 동일한 수집 옵션을 적용하여 최소 엔트로피를 추정할 때마다 유사한 최소 엔트로피로 추정됨을 확인하였다. 본 절에서는 한 번의 실험으로 추정한 GetSystemTime의 최소 엔트로피와 3장에서 휴리스틱 분석으로 추정한 엔트로피를 비교하는 과정을 수집 방법에 따라 진행한다. 이를 통해 선정한 최소 엔트로피 추정 알고리즘에 대한 타당성을 검증하고, GetSystemTime의 엔트로피 추정 방법을 제시한다.

1. '랜덤한 수집 간격' 방법으로 수집한 경우

Table 6.에 있는 수집 시간 간격과 샘플 개수를 적용하여 주어진 시간 범위 내에서 랜덤하게 선택된 시간 간격으로 GetSystemTime을 수집하였다. 그리고 각 옵션에 따라 수집한 데이터에서 S[14] 자리(wMilliseconds의 최하위 바이트)의 값을 추출하고, 최소 엔트로피 추정 알고리즘으로 최소 엔트로피를 추정하였다. Fig. 15.는 수집 옵션을 다음과 같이 적용하였을 때 추정한 최소 엔트로피를 나타낸다.

- 수집 시간 간격(ms)
  : 10, 20, 30, 40, 90, 210, 400
- 수집할 샘플 개수
  : 1,000, 2,000, 5,000, 7,000, 10,000, 50,000

Fig. 15.에서 보이는 바와 같이, 최대 수집 시간 간격이 40 ms 이상으로 주어지면, 수집 시간 간격과 수집한 샘플 개수와는 상관없이 실험을 통해 추정된 GetSystemTime의 최소 엔트로피가 거의 유사함을 알 수 있다. 또한, 적어도 2,000 개의 샘플을 수집하여 S[14] 자리의 값을 추출한 데이터를 최소 엔트로피 추정 알고리즘에 사용하면, 의미 있는 최소 엔트로피가 추정된다. 이로 인해, 샘플 개수가 1,000 개인 실험 결과는 의미가 없다고 판단할 수 있다.

Fig. 16.은 GetSystemTime에 대한 휴리스틱 분석으로 추정한 엔트로피(Table 5.)와 실험으로 추정한 최소 엔트로피를 비교한 결과로, 세 가지 사항을 알 수 있다.

첫 번째로는 최대 수집 시간 간격이 10 ms인 경우 휴리스틱 분석 결과와 실험 결과의 엔트로피 차이가 2 비트인 것이다. 이는 3장의 휴리스틱 분석에서 10 ms 범위 내에서 매번 랜덤하게 수집한 GetSystemTime이 카운터의 특성을 가질 수 있다고 판단하여 보수적으로 엔트로피를 추정하였기 때문

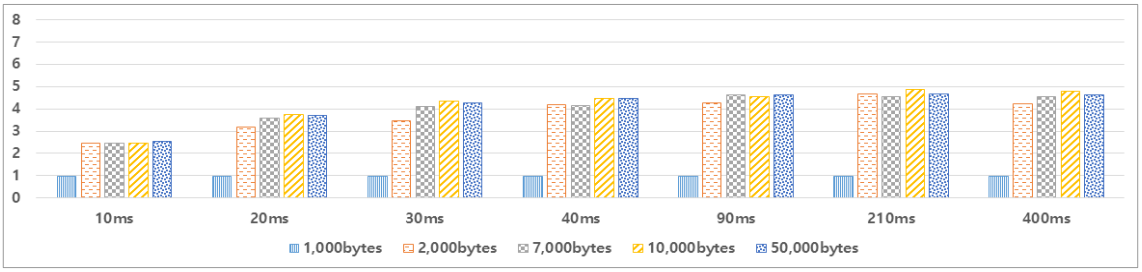

Fig. 15. The estimated min-entropy of GetSystemTime when collecting data at 'random time intervals' method (X-axis: given max collection time interval, Y-axis: min-entropy)

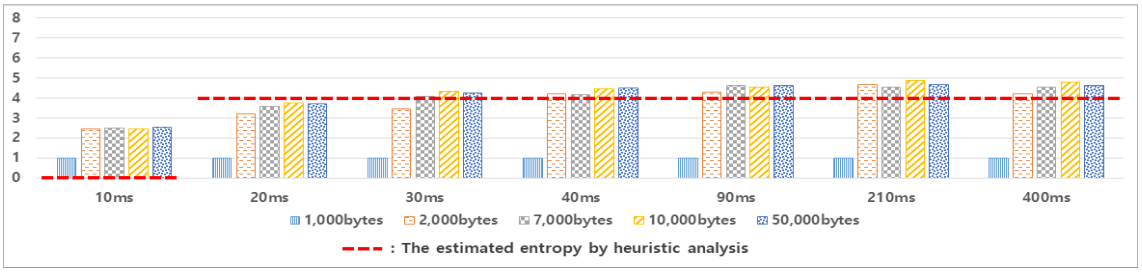

Fig. 16. The comparison of the estimated min-entropy by experiment and the estimated entropy by heuristic analysis when collecting data at 'random time intervals' method (X-axis: given max collection time interval, Y-axis: min-entropy)

에 나온 결과로 판단된다.

두 번째는 최대 수집 시간 간격이 20 ms ~ 30 ms인 경우, 실험 결과가 휴리스틱 분석 결과보다 대부분 낮다는 것이다. 이는 해당 수집 시간 간격 범위 내에서 매번 랜덤하게 수집한 GetSystemTime이 카운터 특성을 가져 엔트로피 추정 알고리즘이 이 특성을 검출한 결과로 판단된다.

세 번째는 40 ms 이상의 시간 간격으로 수집한 경우, 대부분의 실험 결과가 휴리스틱 분석으로 추정한 엔트로피보다 높지만, 휴리스틱 분석 결과를 잘 따르고 있다는 것이다. 그리고 이를 통해 선정한 추정 알고리즘이 타당함을 알 수 있다. 또한, 엔트로피를 추정할 때 샘플 개수에 영향을 덜 받기 때문에, 동일한 수집 시간 옵션을 적용할 때 보다 적은 샘플 개수로도 엔트로피 추정이 가능함을 알 수 있다.

Fig. 16.에서의 결과를 통해 GetSystemTime에 대한 엔트로피 추정 방법을 다음과 같이 제안한다.

- 적어도 2,000 개의 샘플을 수집하여 엔트로피 추정 알고리즘의 입력으로 사용한다.
- GetTickCount에서와 동일하게, 변동률과 예기치 않은 오류가 있는 실험 환경을 고려하여 휴리스틱 분석으로 추정한 엔트로피와 실험으로 추정한 최소 엔트로피 중에서 최솟값을 랜덤한 시간 간격으로 수집한 GetSystemTime의 최소 엔트로피로 추정한다[3].

$$Min\,Entropy = \min\{Entropy_{Heuristic}, Entropy_{\mathrm{Exp}}\}$$

2. '일정한 수집 간격' 방법으로 수집한 경우

Table 6.에 있는 수집 옵션을 적용하여 각각의 수집 시간 간격으로 일정하게 GetSystemTime을 수집하고, 각 옵션에 따라 수집한 데이터에서 S[14] 자리의 값을 추출하여 최소 엔트로피를 추정하였다.

3 장에서 진행한 일정한 수집 시간 간격일 때의 휴리스틱 분석 결과를 보면, GetSystemTime의 추정 엔트로피는 항상 0 비트이다. Fig. 17.은 실험 결과와 휴리스틱 분석 결과를 비교한 것으로, 수집 옵션을 다음과 같이 적용하였다.

- 수집 시간 간격(ms) : 60, 170, 290, 400
- 수집할 샘플 개수 : 4,000, 10,000, 50,000

GetTickCount에서와 동일하게, 충돌 추정과 압축 추정 알고리즘이 모두 풀 엔트로피로 추정하는 경우, 실험으로 추정한 엔트로피가 휴리스틱 분석으로 추정한 0 비트 엔트로피와 같다고 여긴다. 또한, 일

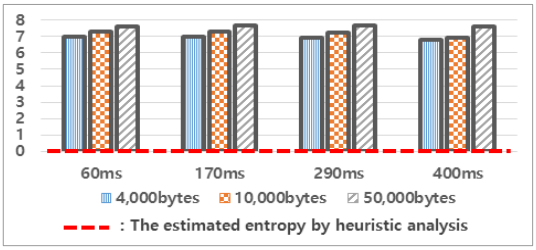

Fig. 17. The comparison of the estimated min-entropy by experiment and the estimated entropy by heuristic analysis when collecting data at 'regular time intervals' method (X-axis: given max collection time interval, Y-axis: min-entropy)

정한 시간 간격으로 수집한 GetSystemTime의 엔트로피를 휴리스틱 분석 결과와 실험 결과 중에서 최솟값으로 추정[3]한다. 따라서 이 경우 항상 0 비트의 최소 엔트로피를 가진다.

V. 제안하는 시간 종속 잡음원에 대한 엔트로피 평가 방법

GetTickCount와 GetSystemTime에 대한 휴리스틱 분석 결과와 실험적 분석 결과를 기반으로 하여 Fig. 18.에서 나타난 바와 같이 시간 종속 잡음원에 대한 엔트로피 평가 방법을 제안한다.

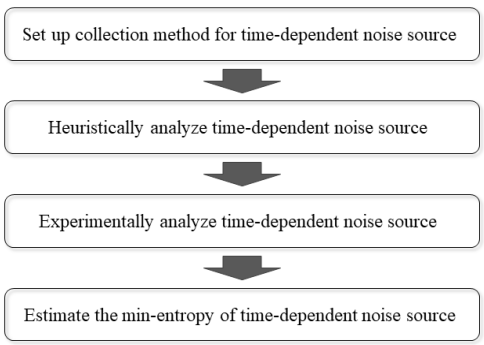

Fig. 18. Proposed entropy evaluation process for time-dependent noise sources

1. 시간 종속 잡음원의 수집 방법 설정

수집 환경을 고려하여 수집 시간 간격과 엔트로피 추정에 사용할 샘플 개수를 설정한다. 2 장에서 살펴본 바와 같이, 시간 종속 잡음원에 대한 수집 방법은 '랜덤한 수집 간격' 방법과 '일정한 수집 간격' 방법이 있다. 첫 번째 방법은 설정한 수집 시간 간격을 최대 수집 간격으로 설정하여 0 ms ~ 최대 수집 시간 간격까지인 범위 내에서 매번 랜덤하게 선택된 시간 간격으로 잡음원을 수집하는 것이다. 두 번째 방법은 설정한 수집 시간 간격으로 일정하게 잡음원을 수집하는 것이다.

본 논문에서는 시간 종속 잡음원을 '랜덤한 수집 간격' 방법을 적용하여 수집하는 것을 권장한다. 그 이유로는 3 장과 4 장에서 진행한 GetTickCount와 GetSystemTime의 분석 결과에서 알 수 있듯이, 랜덤한 수집 시간으로 수집한 경우의 시간 종속 잡음원의 추정 엔트로피가 일정한 시간 간격으로 수집한 경우보다 높기 때문이다. 그뿐만 아니라 암호 시스템과 암호모듈에서 난수를 생성하기 위해 엔트로피를 수집할 때, 동작되는 환경으로 인해 최소 시간 간격 이상인 랜덤한 시간 간격으로 수집할 것으로 간주할 수 있다. 이는 '랜덤한 수집 간격' 방법이 암호 시스템과 암호모듈의 실제 동작 환경과 유사하다고 여길 수 있기 때문이다.

2. 휴리스틱 분석

시간 종속 잡음원에 대한 구조를 분석하고, 이를 기반으로 하여 휴리스틱 분석을 진행한다. 휴리스틱 분석은 본 논문에서 설명한 방법과 유사하게 최적의 바이트 위치와 수집 방법을 고려하면서 진행한다.

3. 실험적 분석

1 단계에서 설정한 수집 방법을 적용하여 시간 종속 잡음원을 수집하고, 2 단계에서 분석한 최적의 바이트 위치의 값을 추출한다. 그런 다음 최소 엔트로피 추정 방법으로 최소 엔트로피를 추정하고, 추정되는 최소 엔트로피의 변동성이 크지 않음을 보이기 위해 동일한 수집 방법을 적용한 반복 실험을 진행한다. 이때의 최소 엔트로피 추정 방법은 본 논문에서 사용한 세 가지의 추정 알고리즘(최빈값 추정, 충돌 추정, 압축 추정)을 사용하는 것을 권장한다. 이 외의 최소 엔트로피 추정 방법이 소프트웨어 잡음원에 적합하다고 여겨지면 사용해도 되지만, 적절한 근거를 제시해야 한다.

4. 최소 엔트로피 추정

최소 엔트로피의 변동성이 크지 않고 실험 결과가 휴리스틱 분석 결과를 잘 따르면, 실험으로 추정한 최소 엔트로피와 휴리스틱 분석으로 추정한 엔트로피

중에서 최솟값을 설정한 수집 방법이 적용된 시간 종속 잡음원의 최소 엔트로피로 추정한다.

$$Min\,Entropy = \min\{Entropy_{Heuristic}, Entropy_{\mathrm{Exp}}\}$$

그리고 2, 3 단계에서 진행한 분석 내용을 최소 엔트로피 추정 근거로 제시한다.

VI. 제안하는 엔트로피 평가 방법의 활용 방안

본 장에서는 NIST SP 800-90B[1]의 선택 사항인 잡음원의 편향성(bias)을 줄이고 엔트로피 비율을 높이는 Conditioning Component를 사용하는 방법을 제시한다. 그리고 Conditioning Component에 제안하는 엔트로피 평가 방법을 활용하는 방법을 제시한다.

6.1 Conditioning Component

Conditioning Component는 Fig. 19.에서 보이는 바와 같이 구성되어 있다. Conditioning Component의 입력과 출력의 비트 크기는 각 n_{in}과 n_{out}로 표현한다. 잡음원의 각 샘플을 연접하여 크기가 n_{in} 비트인 입력을 구성하고, 입력의 엔트로피를 h_{in}으로 표현한다. 이때, h_{in}은 n_{in} 비트의 입력을 구성하는데 필요한 잡음원 샘플의 개수에 의존한다. 즉, n_{in} 비트의 입력을 구성하는데 w 개의 샘플이 필요하고 잡음원의 엔트로피가 h 비트라면, h_{in} 비트는 $w \times h$ 비트로 추정된다. Conditioning Component는 결정론적 함수이기 때문에 출력의 엔트로피 h_{out}은 최대 h_{in} 비트이지만, 엔트로피 저하가 있을 수 있다. 또한, h_{out}은 Conditioning Component의 내부 요소에 대한 변수(narrow internal width)인 nw와 사용되는 Conditioning 함수에 따라 달라진다. SP 800-90B에서는 FIPS와 NIST에서 승인한 암호학적 Conditioning 함수를 사용하는 경우와 그렇지 않은 경우로 나누고, 각각의 경우에 따라 출력의 엔트로피 h_{out} 비트를 추정한다.

본 장에서는 NIST에서 승인한 암호학적 알고리즘

Fig. 19. Entropy of the Conditioning Component[1]

을 사용하여 h_{out} 비트를 추정하고자 한다. Table 12.는 SP 800-90B에 작성된 FIPS와 NIST에서 승인한 암호학적 Conditioning 함수와 각각에 대한 nw와 n_{out}을 나타낸다. 이를 이용하여 SP 800-90B에서는 출력 엔트로피 h_{out}을 Output_Entropy(n_{in}, n_{out}, nw, h_{in})로 추정한다. Output_Entropy 함수에 대한 설명은 SP 800-90B의 3.1.5.1.2 절에 있다.

6.2 제안하는 엔트로피 평가 방법의 활용

본 절에서는 제안하는 엔트로피 평가 방법을 사용하여 시간 종속 잡음원인 GetTickCount의 엔트로피를 추정하고, SP 800-90B의 Conditioning Component를 사용하여 GetTickCount의 편향성을 줄이고 엔트로피 비율을 높이고자 한다.

먼저, GetTickCount의 엔트로피를 본 논문에서 제안하는 방법으로 추정하기 위해, 최대 수집 시간이 70 ms로 주어진 시간 범위 내에서 매번 랜덤한 시간 간격으로 5,000 개의 GetTickCount를 수집하도록 수집 방법을 설정하였다. 그런 다음, 3 장에서와 같은 방법으로 GetTickCount에 대한 구조 분석과 휴리스틱 분석을 진행하여 2 비트의 엔트로피로 추정한다. 그리고 설정한 수집 방법으로 GetTickCount를 수집하여 4 장에서와 같은 실험 방법으로 최소 엔트로피를 추정하고, 추정되는 엔트로피의 변동성이 크지 않음을 보이고자 동일한 수집 옵션을 적용한 실험을 5 번 반복 진행하였다. 실험을 통해 추정된 최소 엔트로피는 2.36 비트이고, 이때의 변동폭은 2.34 ± 0.05로, 4.73 % 이내의 최소 엔트로피 차이를 가진다. 실험 결과가 휴리스틱 분석 결과를 잘 따르고 최소 엔트로피의 변동성이 크지 않으므로, 제안하는 엔트로피 추정 방법을 적용하면 최대 수집 시간이 70 ms로 매번 랜덤하게 수집된 GetTickCount의 최소 엔트로피는 2 비트($= \min\{Entropy_{Heuristic}, Entropy_{\mathrm{Exp}}\}$ $= \min\{2\text{ 비트}, 2.36\text{ 비트}\}$)로 추정된다.

이와 같이 추정한 GetTickCount의 엔트로피 비율을 높이고자 Fig. 20.에서와 같이 SP 800-90B의 Conditioning Component를 사용하였다. 그리고 FIPS와 NIST에서 승인한 암호학적 Conditioning 함수를 적용하기 위해, 해시 함수인 SHA-256[7]을 사용하였다. 이 경우에는, Table 12.에서와 같이, $nw = n_{out} = 256$이다. 본 논문에서는 h_{in}을 512로 고정하였다. 이는 일반적으로 출력의 크기 n_{out}의 2 배 이

Table 12. The list of vetted Conditioning Components, the narrowest internal width(nw) and the output length(n_{out})[1]

Conditioning function	Narrowest Internal width(nw)	Output Length(n_{out})
HMAC	hash-function output size	hash-function output size
CMAC	AES block size = 128	AES block size = 128
CBC-MAC	AES block size = 128	AES block size = 128
Hash Function	hash-function output size	hash-function output size
Hash_df	hash-function output size	hash-function output size
Block_Cipher_df	AES key size	AES key size

상인 h_{in} 비트의 엔트로피가 Conditioning 함수에 입력되어야 하기 때문이다. GetTickCount의 최소 엔트로피를 2비트로 추정하였으므로, h_{in}=512비트가 되기 위해 총 256개의 GetTickCount 샘플이 필요하다. 그리고 GetTickCount 샘플의 크기는 32비트이므로, 입력의 크기 n_{in}은 $32 \times 256 = 8{,}192$로 고정하였다. 따라서 256개의 GetTickCount를 연접하여 $n_{in} = 8{,}192$비트인 입력을 구성하였고, 이를 입력 받은 SHA-256은 크기가 $n_{out} = 256$비트인 해시값을 출력하였다. NIST SP 800-90B에 따르면, 이 출력의 엔트로피 h_{out}는 Output_Entropy($n_{in} = 8{,}192$, $n_{out} = 256$, $nw = 256$, $h_{in} = 512$)로 추정된 256비트이다.

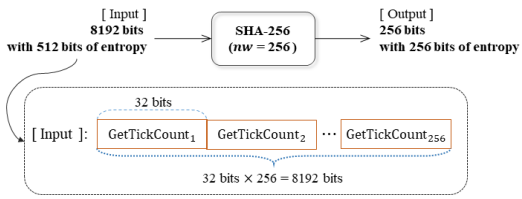

Fig. 20. Using Conditioning Component for GetTickCount

VII. 결 론

암호모듈 검증제도(CMVP)[4]에서는 난수발생기의 입력으로 112비트 이상의 엔트로피를 요구하므로, 이를 입증하기 위한 잡음원의 엔트로피 평가 방법에 대한 연구가 필요하다. 그러나 주요 국외 표준문서인 SP 800-90B[1], AIS.31[8]의 엔트로피 평가 방법을 소프트웨어 잡음원에 적용하는 것이 어렵고, 소프트웨어 잡음원별 엔트로피에 대한 정량적 평가 방법과 그 근거가 부족하다. 이는 소프트웨어 잡음원이 운영체제와 수집 시간에 의존적이고 대량의 데이터 수집이 어렵다는 특징을 가지고 있기 때문이다. 따라서 본 논문에서는 소프트웨어 잡음원에 적합한 엔트로피 평가 방법을 제안하였다. 먼저, 소프트웨어 잡음원의 특징을 반영하여 소프트웨어 잡음원에 대한 분석을 진행할 때 고려해야 하는 사항을 제시하였다. 그리고 윈도우 운영체제에서 수집 가능한 소프트웨어 잡음원 중에서 시간 종속 잡음원인 GetTickCount와 GetSystemTime을 분석 대상으로 선정하여, 이에 대해 휴리스틱 분석과 실험적 분석을 진행하였다. 이를 통해, 주어진 시간 범위 내에서 랜덤하게 선택된 시간 간격으로 시간 종속 잡음원을 수집하고, 휴리스틱 분석으로 추정한 엔트로피와 실험으로 추정한 엔트로피 중에서 최솟값을 잡음원의 최소 엔트로피로 추정하는 것이 적절함을 확인하였다. 또한, 본 논문에서는 휴리스틱 분석과 실험 분석의 결과를 기반으로 하여 시간 종속 잡음원의 수집 방법과 엔트로피 추정 방법을 제안하였다. 그리고 제안하는 방법을 NIST SP 800-90B[1]의 Conditioning Component에 활용하여 잡음원의 편향성을 줄이고 엔트로피 비율을 높이는 방법을 제시하였다.

본 연구의 결과가 소프트웨어 잡음원의 엔트로피에 대한 정량적 평가의 어려움을 해결하고, 국내외적으로 부족한 소프트웨어 잡음원의 엔트로피 추정 근거를 제공하는 기반이 될 것으로 기대한다. 또한, 한국 암호모듈 검증제도의 검증필 암호모듈 목록을 참조하면 국내에서 개발되는 대부분의 암호모듈은 소프트웨어 기반이므로, 본 연구의 결과가 국내 소프트웨어 암호모듈의 개발과 안전성 평가에 활용될 것으로 기대한다.

References

[1] M.S. Turan, E. Barker, J. Kelsey, K.A. McKay, M.L. Baish, and M. Boyle, "Recommendation for the Entropy Sources Used for Random Bit Generation," NIST Special Publication 800-90B, Jan. 2018.

[2] Sang Yun Han, Seogchung Seo, Yongjin Yeom, and Yewon Kim, "Entropy Evaluation Algorithms for Noise Sources in Software Environments," TTAK.KO-12.0306/R1, Jun. 2018.

[3] Yewon Kim, "A study on the Entropy Analysis for Time-Dependent Noise Sources and the Parallel Implementation of the Randomness Tests," Master's thesis, Kookmin University, Aug. 2017.

[4] NIST, "Implementation Guidance for FIPS 140-2 and the Cryptographic Module Validation Program," Jan. 2018.

[5] MSDN, "Windows 10 API sets" https://msdn.microsoft.com/en-us/library/windows/desktop/dn764993(v=vs.85).aspx, accessed Aug. 13, 2018.

[6] Yewon Kim, Ju-Sung Kang, and Yongjin Yeom, "A study on an estimation method for noise sources related to time of Windows operating system," Conference on Information Security and Cryptography 2016 Winter, Dec. 2016.

[7] NIST, "Secure Hash Standard (SHS)," FIPS PUB 180-4, Mar. 2012.

[8] W. Killmann and W. Schindler, "A Proposal for : Functionally classes and evaluation methodology for true (physical) random number generators," BSI, Sep. 2001.

[9] ISO, "Information technology -- Security techniques -- Test and analysis methods for random bit generators within ISO/IEC 19790 and ISO/IEC 15408," ISO/IEC DIS 20543, Jan. 2018.

[10] M.S. Turan, E. Barker, J. Kelsey, K.A. McKay, M.L. Baish, and M. Boyle, "Recommendation for the Entropy Sources Used for Random Bit Generation(Second DRAFT)," NIST Special Publication 800-90B, Jan. 2016.

〈저자소개〉

김 예 원 (Yewon Kim) 학생회원
2015년 8월: 숙명여자대학교 수학과 학사
2017년 8월: 국민대학교 일반대학원 금융정보보안학과 석사
2017년 9월~현재: 국민대학교 일반대학원 금융정보보안학과 박사과정
〈관심분야〉 암호구현, 난수성 분석 및 평가, 병렬 프로그래밍

염 용 진 (Yongjin Yeom) 종신회원
1991년 2월: 서울대학교 수학과 학사
1994년 2월: 서울대학교 수학과 석사
1999년 2월: 서울대학교 수학과 박사
2000년 4월~2012년 2월: ETRI 부설연구소 책임연구원/팀장
2006년 12월~2007년 12월: Columbia 대학교 방문 연구원
2012년 3월~현재: 국민대학교 과학기술대학 정보보안암호수학과 부교수
2013년~현재: 국민대학교 BK21+ 미래 금융정보보안 인력양성사업단 교수
〈관심분야〉 암호구현 및 분석, 보안시스템 평가