ICMC 2017

Entropy Estimation Methods for SW Environments in KCMVP

NSR: Seogchung Seo, Sangwoon Jang
Kookmin University: Yewon Kim, Yongjin Yeom

Contents

Brief Introduction to KCMVP (1/3)

KCMVP (Korea Cryptographic Module Validation Program)

- Since 2005
- Validating security and implementation conformance of a CM for protecting sensitive information in public/ governmental institutes
- Based on KS X ISO/IEC 19790:2015, 24759:2015
- CAVP is conducted in KCMVP process

Standard

- Security Requirements: KS X ISO/IEC 19790:2015
- Test Requirements: KS X ISO/IEC 24759:2015
- Begin to apply 2015 version since June 2016
- Approved Algorithms: ISO/IEC, KS, TTA

Brief Introduction to KCMVP (2/3)

Approved Algorithms in KCMVP

Types		Algorithms
Block cipher		ARIA-128/192/256, LEA-128/192/256, SEED, HIGHT
Hash function		SHA-224/256/384/512
MAC	Hash-based	HMAC(SHA-224/256/384/512)
	Block cipher-based	GCM(GMAC) CCM, CMAC
RBG	Hash_DRBG	SHA-224/256/384/512
	HMAC_DRBG	HMAC(SHA-224/256/384/512)
	CTR_DRBG	ARIA-128/192/256, LEA-128/192/256, SEED, HIGHT
Key establishment		DH: (2048, 224), (2048, 256) ECDH: P-224/256, B-233/283, K-233/283
Public key encryption		RSAES: |n|=2048/3072
Digital signature		RSA-PSS, KCDSA, ECDSA, EC-KCDSA

LEA: A 128-bit Block Cipher for Fast Encryption on Common Processors, WISA 2013, LNCS 8267, 2014.
HIGHT: A New Block Cipher Suitable for Low-Resource Device, CHES 2006, LNCS 4249, 2006.

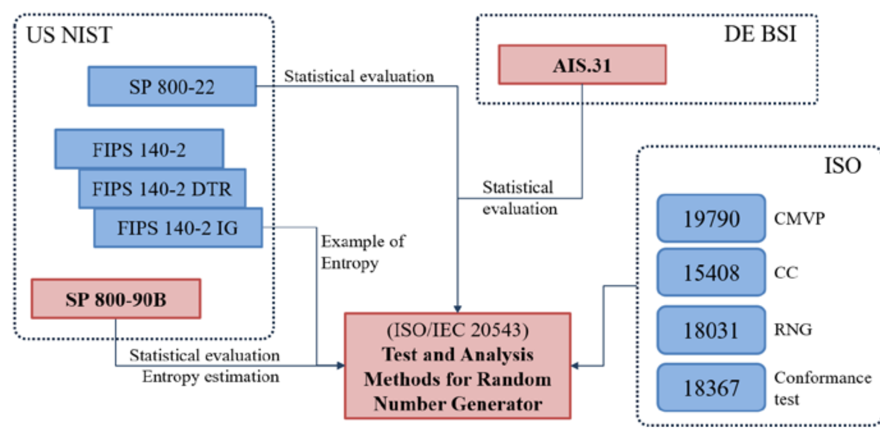
Brief Introduction to KCMVP (3/3)

- Current Validation Statistics in KCMVP
  - More than 160 modules have been validated
    - SW modules are dominant in the validation list
    - Almost security level 1
- Preferred Environments
  - Windows > Linux/Unix > Java > Mobile(Android, iOS)
- Most Frequently used approved algorithms in KCMVP
  - Symmetric-key > Hash > MAC > RBG > Public-key encryption > Digital Signature > KE

Why is Entropy Analysis important?

- Modern Cryptosystems Depends on the Security of Underlying Key
- Security of Key Depends on the Seed Value in DRBG

Not enough entropy in collected noise sources
→ weakness in whole security systems



Noise sources for RBG seed

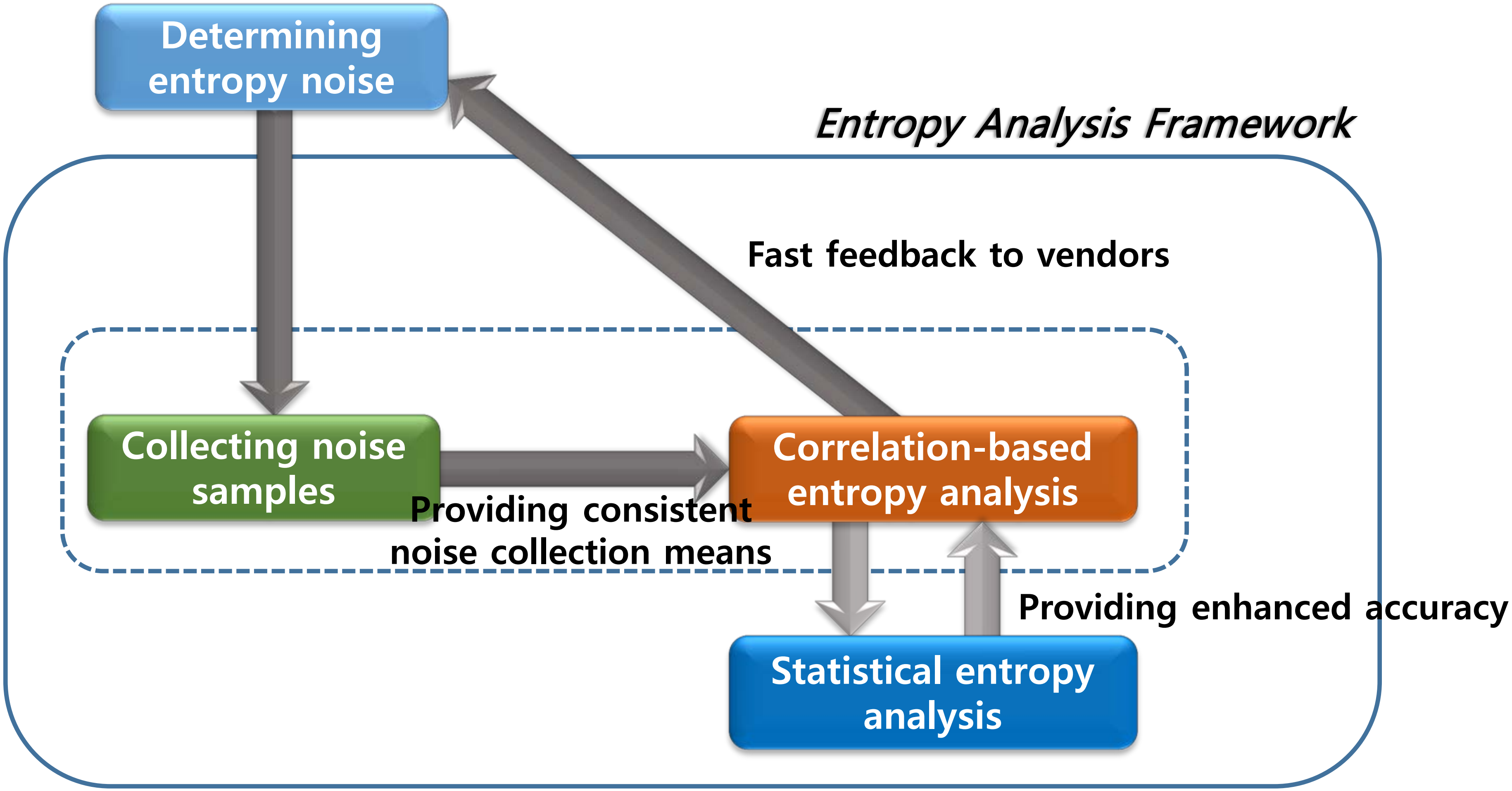
Entropy Estimation in S/W Environments

- Characteristics of Noise Source in SW Environments
  - Biased entropy distribution
    - Difficult to expect uniform distribution or IID
  - Dependence among bytes in a source
    - Same data can be repeated
  - Sample size btw several bytes and several hundreds bytes
  - Depends on how to collect and types of OSs
    - Collection interval affects the entropy
  - Generation rate is not consistent
    - Ex) mouse events, key board events and so on

Current methods are suitable for HW noise sources.
Thus, entropy evaluation method for SW noise sources is necessary!

Entropy Analysis Framework

- Overall Structure of Entropy Analysis Framework in KCMVP



- *Under progress of TTA standardization ("Entropy Evaluation Algorithms for Noise Sources in Software Environments")*

Correlation-based Entropy Analysis

- Aiming at
  - fast, but moderately accurate entropy analysis for rapid feedback
  - Reducing efforts for gathering noise samples
    - At least, 256 samples required for a noise source
- Format of noise sample

```
  1 eca2dcda75e3245244ef08e31fa31a19eda9cbb7f5c4ac2cce76f63c69f0cb5e
  2 2b23c6cb08b4a07217bc3cf05d006acc986c180fd50078f4c2af0ca7cd1715b9
  3 26350b469e1f44291323e2b27d4707b1d46860535a14c174f0eda7f49c5d1683
  4 bbe37eb9dfec2759cb934f9ce5719f43418ef92e7b30bd8361637028bee550a7
  5 397a1abffcfb4ba1a2e5be133b60703a357d62ab0c5d5cae89c3518d5d46c5ff
  6 ac2d7c34a87ced1f6fc8e278231b2f095c7adffd8dba74bb3af0e951369a7fea
  7 02321c33da0d358a949949098683f9f5b2ae078b683a19c48e6f5b807b772f6f
  8 5f7c423ff526b5b484a46baa6c2d9cd5beec3345373571dbd45aa79e92be2160
  9 313270138b6eb77b739214fa871a74f5e20bd1353312bf3653b30aa27f89e590
...
250 cced1596ecf22a001d2b06faece71a0b7d7238c0c82176bc938ca8ee3389aa05
251 60aeb39cb79d045e411174d3c296e1e3231e23089d38e86b65aea79038b360f2
252 20717db62d4f2e72a4eb18ef5f7685d6590591ab6f1e021aab64aa0a6a527aa9
253 a9e2b791ed9d078f38ade9c60007b88367c23d35fb135eecee2b9245449e305e
254 719832104feffb2cca9e3436cae7c617c88a8dc807b2f3e54d9f23074916812b
255 a6c8a6067fbd721dce6db6ccc9897864de3f32ae5d10a161786fbc21ecaefd2d
256 3d430457d006048becb7187957c53b1ad5ae981ef666e286561e8dea6819a549
```

\# of samples

\# of byte columns

Correlation-based Entropy Analysis

- Process of CEA
  - Distribution-based filtering
    - Byte sample distribution, 0/1's distribution
  - Byte-oriented entropy analysis
    - Maximum entropy is 8-bit on a byte column
    - Applying Shannon, Min entropy
    - Can be extended to use other statistical entropy analysis
  - Pearson correlation-based entropy reassessment

Distribution-based Filtering → Byte-oriented Entropy analysis → Correlation Computation among bytes → Entropy Reassessment

Distribution-based Filtering	Byte-oriented Entropy analysis	Correlation Computation among bytes	Entropy Reassessment
• Byte value distribution • 0/1's distribution	• Shannon entropy • Min entropy	• Computing Pearson correlation among byte columns	• Entropy reassessment

Correlation-based Entropy Analysis

- Distribution-based Filtering
  - Byte values: 8, 16, 32, 64, 128
  - 0/1's values:

Sample #	0-th byte	1-th byte	2-th byte	3-th byte	4-th byte
0	0F	01	32	6C	B7
1	F2	02	F4	7A	F2
2	CC	01	AB	4F	39
…	…	…	…	D3	73
255	EF	02	97	46	96
#Dist	128	5	150	154	180

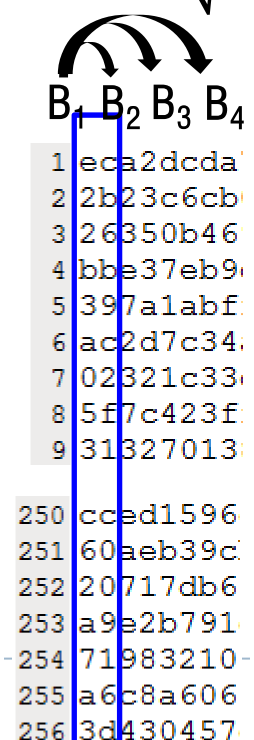
- Byte-oriented Entropy Analysis
  - Individual entropy computation
  - Computing Shannon entropy
    - $\sum_{i=0}^{255}(-P_i \log_2 P_i)$
  - Computing Min entropy
    - $-\log_2(\max(p_0, \dots, p_{255}))$
  - Can be extended to other entropy computation methods

Sample#	0-th byte	2-th byte	3-th byte	4-th byte
0	0F	32	6C	B7
1	F2	F4	7A	F2
2	CC	AB	4F	39
…	…	…	D3	73
255	EF	97	46	96
#Dist	128	150	154	180
Entropy	e_0	e_1	e_2	e_3

Correlation-based Entropy Analysis

- Computing Correlation among Byte Columns
  - $|cor_{a,b}| \leq 1$, correlation btw a-th and b-th byte columns, $cor_{a,a} = 0$

$$cor_{a,b} = \frac{\sum_{i=0}^{255}(B_{a,i} - \overline{B_a})(B_{b,i} - \overline{B_b})}{\sqrt{\sum_{i=0}^{255}(B_{a,i} - \overline{B_a})^2 \times \sum_{i=0}^{255}(B_{b,i} - \overline{B_b})^2}}$$



$cor_{1,1}$	$cor_{1,2}$	$cor_{1,3}$	$cor_{1,4}$
$cor_{2,1}$	$cor_{2,2}$	$cor_{2,3}$	$cor_{2,4}$
$cor_{3,1}$	$cor_{3,2}$	$cor_{3,3}$	$cor_{3,4}$
$cor_{4,1}$	$cor_{4,2}$	$cor_{4,3}$	$cor_{4,4}$

k X k correlation table
(in this case, 4 X 4)

Correlation-based Entropy Analysis

- Entropy Reassessment
  - Basic Idea
    - *Total entropy needs to be bigger than individual byte column's entropy*

$$\left(\sum_{i=0}^{k} e_i\right) \geq e_j, where\ 0 \leq j \leq k$$

  - General equation

Reassessed entropy = original entropy x Reducing factor

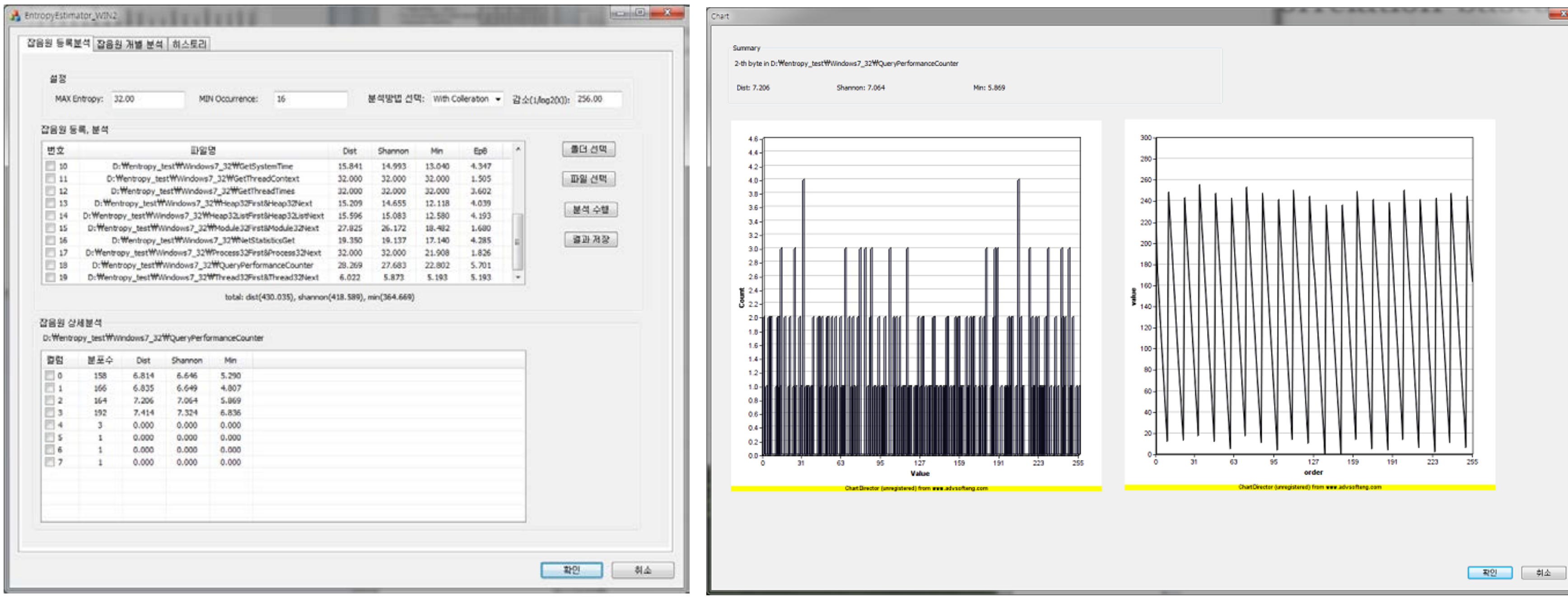
$$e_i' = e_i\left[1 - \frac{1}{k}\left(\sum_{j=0}^{k} c_{i,j}\right)\right] where\ i \neq j\ and\ k\ is\ \#of\ byte\ columns$$

$$e_1' = e_1\left[1 - \frac{1}{4}(|cor_{12}|+|cor_{13}|+|cor_{14}|)\ \right]$$
$$e_2' = e_2\left[1 - \frac{1}{4}(|cor_{21}|+|cor_{23}|+|cor_{24}|)\ \right]$$
$$e_3' = e_3\left[1 - \frac{1}{4}(|cor_{31}|+|cor_{32}|+|cor_{34}|)\ \right]$$
$$e_4' = e_4\left[1 - \frac{1}{4}(|cor_{41}|+|cor_{42}|+|cor_{43}|)\ \right]$$

Entropy reassessment example (k=4)

Correlation-based Entropy Analysis

- Developed SW – Entropy Estimator V0.1



Correlation-based Entropy Analysis

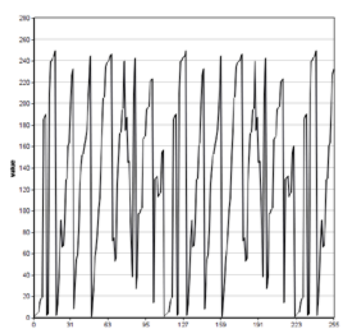
- ## Experimental Result
  - ### Linux Kernel 2.6 64-bit

D:₩Development₩Fasoo_entropy₩Linux_64₩pthread_self	14.534	14.200	11.356	5.678
D:₩Development₩Fasoo_entropy₩Linux_64₩random	32.000	32.000	32.000	5.797
D:₩Development₩Fasoo_entropy₩Linux_64₩schedstat	32.000	32.000	32.000	5.676
D:₩Development₩Fasoo_entropy₩Linux_64₩slabinfo	32.000	30.464	17.025	1.892
D:₩Development₩Fasoo_entropy₩Linux_64₩stat	22.366	20.643	15.320	3.830
D:₩Development₩Fasoo_entropy₩Linux_64₩time	7.651	7.563	7.000	7.000
D:₩Development₩Fasoo_entropy₩Linux_64₩uptime	0.000	0.000	0.000	0.000
D:₩Development₩Fasoo_entropy₩Linux_64₩uuidgen	0.000	0.000	0.000	0.000
D:₩Development₩Fasoo_entropy₩Linux_64₩vmstat	32.000	32.000	32.000	5.442

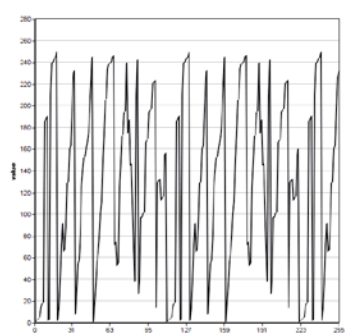
Without entropy reassessment

D:₩Development₩Fasoo_entropy₩Linux_64₩pthread_self	14.438	14.106	11.281	5.641
D:₩Development₩Fasoo_entropy₩Linux_64₩random	32.000	32.000	32.000	5.535
D:₩Development₩Fasoo_entropy₩Linux_64₩schedstat	32.000	32.000	32.000	5.312
D:₩Development₩Fasoo_entropy₩Linux_64₩slabinfo	32.000	26.316	14.694	1.633
D:₩Development₩Fasoo_entropy₩Linux_64₩stat	15.392	14.206	10.543	2.636
D:₩Development₩Fasoo_entropy₩Linux_64₩time	7.651	7.563	7.000	7.000
D:₩Development₩Fasoo_entropy₩Linux_64₩uptime	0.000	0.000	0.000	0.000
D:₩Development₩Fasoo_entropy₩Linux_64₩uuidgen	0.000	0.000	0.000	0.000
D:₩Development₩Fasoo_entropy₩Linux_64₩vmstat	32.000	32.000	32.000	4.864

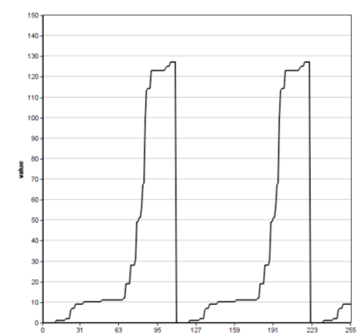
Without entropy assessment



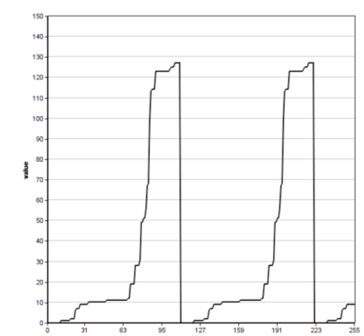
0-th byte column



14-th byte column



1-th byte column



15-th byte column

Correlation-based entropy reassessment prevents entropy overestimation!

Standardization of Entropy Evaluation Algorithms for Noise Sources in Software Environments and its Application

2017.05.17.(Wed.)

Kookmin University

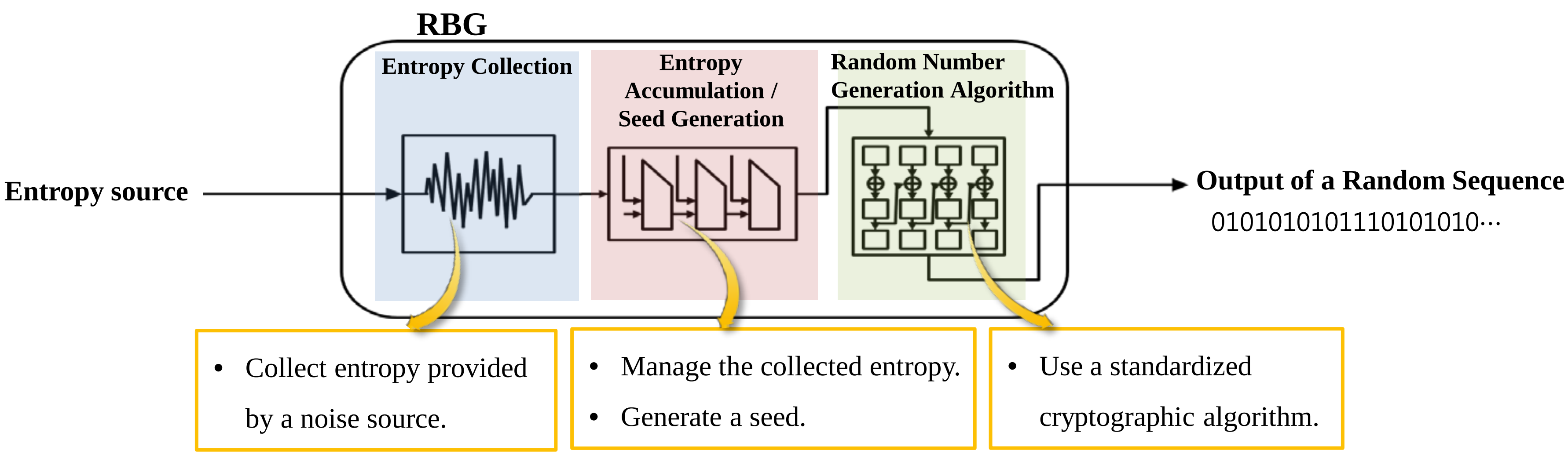
Contents

- Abstract
- Standardization : software noise source evaluation method
- Evaluation method of software noise source

Abstract

Abstract

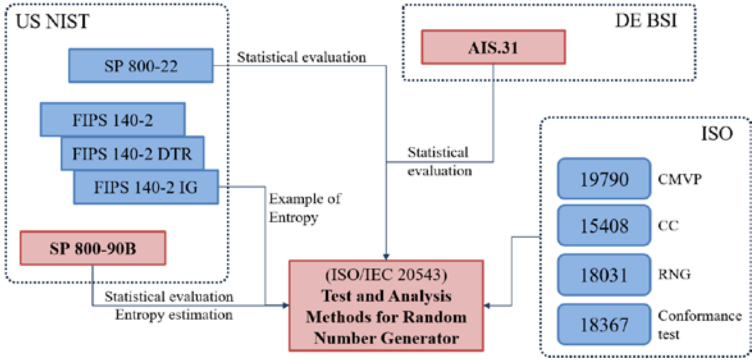
- **What is RBG(Random Bit Generator)?**
  - RGB generates a random bit for the cryptographic systems.
  - Ideally, the generated random bit is expected to be the result of 'coin toss'.
  - It is an essential element in the operation of the cryptographic systems and the cryptographic modules.
  - Required properties: unpredictability, non-bias/uniformity, bit-to-bit independence



- **Vulnerability of Entropy Collection in RBG**
  - **If RBG do not collect enough entropy, the output of a random number is predictable.**

Abstract

- **Primary Documents and Standards for Entropy Evaluation Methods**
  - BSI AIS.31
    - Statistical tests for the entropy sources
    - a basis for validation by CC(Common Criteria for Information Technology Security Evaluation)
  - NIST SP 800-90B (2nd Draft)
    - Design and testing requirements for the entropy sources
    - a basis for validation by CMVP(Cryptographic Module Validation Program)
  - ISO/IEC 20543 (1st CD)
    - International standard evaluation methodology for entropy



[Relationship between ISO/IEC 20543 and the documents on the evaluation of RNG]

Hardware Nosie Source-Based Entropy Evaluation Method!

Abstract

- In Korea, **software cryptographic modules** are mainly developed.
  - Software noise sources are collected to generate random numbers.

[The list of verified cryptographic modules by KCMP]

암호모듈명	검증번호	개발사	모듈형태	검증일	효력만료일
SCCrypto V1.0	CM-122-2021.9	소프트캠프(주)	S/W(라이브러리)	2016-09-22	2021-09-22
MagicKCrypto V1.0.0	CM-121-2021.9	(주)드림시큐리티	S/W(라이브러리)	2016-09-22	2021-09-22
TCLM V1.1	CM-120-2021.8	유넷시스템(주)	S/W(라이브러리)	2016-08-16	2021-08-16
Fasoo Crypto Framework v2.3	CM-119-2021.8	(주)파수닷컴	S/W(라이브러리)	2016-08-16	2021-08-16
MagicCrypto V2.1.0	CM-118-2021.8	(주)드림시큐리티	S/W(라이브러리)	2016-08-01	2021-08-01
SNIPER Crypto Lite V 1.0	CM-117-2021.7	(주)윈스	S/W(라이브러리)	2016-07-06	2021-07-06
K-Crypto V3.2	CM-116-2021.6	킹스정보통신(주)	S/W(라이브러리)	2016-06-02	2021-06-02
nSafer V1.1					
RTCrypto V1.					
SECUI CryptoU					

S/W Cryptographic modules	H/W Cryptographic modules
95	2

Reference : http://www.nis.go.kr/AF/1_7_3_3/list.do

- **Characteristics of software noise source**
  - The S/W noise source depends on the operating system.
  - It is difficult to expect that the S/W noise source has the uniform distribution or it is IID.
    (IID : Independent and Identically Distributed)
  - The sample size of the S/W noise source is several bytes to several hundred bytes, but the collected entropy is low.
  - It is difficult to collect the S/W noise source of data size required by primary entropy evaluation methods.
  - The characteristic of a S/W noise source can be greatly changed according to the collection interval of the S/W noise source.

It is not desirable to apply the entropy evaluation method that is suitable for the hardware noise source directly to the software noise source.
An entropy evaluation method suitable for software noise sources is needed!

Standardization

: software noise source evaluation method

Standardization : software noise source evaluation method

- It is Korean standard and is registered in TTA(Telecommunications Technology Association).

- Title

  : **Entropy Evaluation Algorithms for Noise Sources in Software Environments**
- Purpose

  : This standard specifies evaluation and statistical test algorithms for DRBG in software environments.
- Summary

  : This standard specifies resource collecting methods, statistical test algorithms and entropy estimate algorithms in software environments.

- TTA Homepage(eng) : http://www.tta.or.kr/eng/index.jsp

TTA standard: Entropy Evaluation Algorithms for Noise Sources in S/W Environments

- **Contents of this TTA standard**

1. **Statistical test algorithm**
   - Statistical test algorithm
   - Health test algorithm
2. **Entropy estimate algorithm based on probability theory and information theory**
   - Min-entropy estimate algorithm
   - Shannon-entropy estimate algorithm
3. **Entropy estimate algorithm based on byte correlation**
4. **Test vector**

TTA standard: Entropy Evaluation Algorithms for Noise Sources in S/W Environments

1. Statistical Test Algorithm

- **Statistical test algorithm**
  - T1 ~ T5 of Class P1 in AIS.31 were selected for statistical test using small data.
  - Statistical test algorithms are **the generalized algorithms that applied P-value to T1 ~ T5.**
  - So variable evaluation criterion can be applied these statistical test algorithms.

- **Health test algorithm**
  - Health test algorithms are the algorithms that generalized Repetition Count Test and Adaptive Proportion Test in SP 800-90B.

TTA standard: Entropy Evaluation Algorithms for Noise Sources in S/W Environments

2. Entropy estimate algorithm based on probability theory and information theory

- **Min-entropy estimate algorithm**
  - The most common value estimate, the collision estimate, the Markov estimate and the compression estimate of entropy estimation for Non-IID data in SP 900-90B were selected, since the software noise source will be Non-IID.
  - Min-entropy estimate algorithms are the algorithms that generalized the selected estimate tests.

- **Shannon-entropy estimate algorithm**
  - There are two types of Shannon-entropy estimate algorithms.
  - One is the generalized T8 of Class P2 in AIS.31.
  - The other is the Shannon-entropy estimate test based on mutual information.
    - Reference : Young-Sik Kim, Yongjin Yeom, and Hee Bong Choi, "Online Test Based on Mutual Information for True Random Number Generators", J. Korean Math. Soc. 50, No. 4, 2013.

Evaluation method of software noise source

Evaluation method of software noise source

- Subject : **GetSystemTime, GetTickCount**

Windows 7(32bits)	Noise Source	Sample Size	Function
Time-related noise source	**GetSystemTime**	**16**	**GetSystemTime();**
	GetTickCount	**4**	**GetTickCount();**
	QueryPerformanceCounter	8	QueryPerformanceCounter();
User-related noise source	GetCursorPos	8	GetCursorPos();
OS-related Noise source	GetCurrentThreadId	4	GetCurrentThreadID();
	GetForegroundWindow	4	GetForegroundWindow();
	GetIcmpStatistics	104	GetIcmpStatistics();
	GetIpStatistics	92	GetIpStatistics();
	GetPerformanceInfo	56	GetPerformanceInfo
	GetProcessHeap	4	GetProcessHeap();
	GetTcpStatistics	60	GetTcpStatistics();
	GetUdpStatistics	20	GetUdpStatistics();
	GlobalMemoryStatusEx	64	GlobalMemoryStatusEx();
	HeapList	32	Heap32ListFirst(); Heap32ListNext();
	ProcessList	1728	Process32First(); Process32Next();
	ThreadList	10248	Thread32First(); Thread32Next();

Evaluation method of software noise source

- Subject : **GetSystemTime, GetTickCount**
  - Time-related noise sources in Windows OS

- Evaluation Scenario
  1. **Theoretical Analysis**

     ➔ Propose the evaluation method for each noise source.
  2. **Experimental Analysis**

     ➔ Validate the proposed evaluation method and propose the collection method.

Theoretical Analysis – Heuristic Analysis in CMVP IG

- A heuristic analysis of time-related noise sources in the Cryptographic Module Validation Implementation(CMVP IG)[1] provided by NIST
  - Representation of time = **hh : mm : ss.zzz**
    - **zzz** : The decimal fraction of a second measured up to the third decimal point(ms, milliseconds). The most variable value.
  - **The estimated entropy of time-related noise source is dependent on the frequency of the measurement.**
    - **If measure at different frequency each time**,
      - The number of the zzz values : 1,000 ➔ **approximately 10 bits of entropy**
      - But it is difficult to measure the time at different frequencies each time.
    - **If measure at a frequency of about 0.5 seconds each time**,
      - The number of the values made out of the second and third "z" : 100
      - The first z has some randomness in it as well.
      - The variability of the zzz values is similar to having 128. ➔ **7 bits of entropy**
      - The CMVP may even accept a claim of 8 bits of entropy in this case if a slightly more sophisticated argument is made to support such a claim.

[1] Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program (Update: 2016.08.01.)

Theoretical Analysis - GetSystemTime, GetTickCount

- Scenarios for Theoretical Analysis
  1. Analyze the structure and characteristics of each noise source.
     - Refer to Microsoft Developer Network(MSDN) provided by Microsoft.
  2. Analyze heuristically the entropy of each noise source.
     - Use **conservatively** the heuristic analysis in CMVP IG.

- Due to the characteristics of software noise source, the characteristic of a software noise source can be greatly changed according to the collection interval.

➡ The analysis of each noise source is performed considering the collection interval as follows.

- **Case of collecting at regular intervals**
- **Case of collecting at random intervals within a given collection interval**

Theoretical Analysis - GetSystemTime

1. Analyze the structure and characteristics of each noise source.

- GetSystemTime is **the current system date and time**.
  - The System time is expressed in Coordinated Universal Time(UTC).
- Sample Size : **16 bytes**
- Collecting function : **GetSystemTime()**
- Components of GetSystemTime

Size	Component
2 bytes	Year (1601~30827)
2 bytes	Month (1~12)
2 bytes	Day of Week (0~6)
2 bytes	Day (1~31)
2 bytes	Hour (0~23)
2 bytes	Minute (0~59)
2 bytes	Second (0~59)
2 bytes	**Milliseconds (0~999)**

- The Most variable component : Milliseconds(ms)
  - **The heuristic analysis is focused on milliseconds(ms).**

Theoretical Analysis - GetSystemTime

2. Analyze heuristically the entropy of each noise source.

- The time is represented as **hh:mm:ss.zzz** that is the same representation as CMVP IG.
  - The heuristic analysis in CMVP IG is **conservatively used** to estimate entropy.

1) Case of collecting at regular intervals

- The number of the zzz values : **1**
  - ✓ Ex) When collecting at 160 ms intervals, always collect zzz value increased by 160.
- Estimated entropy : **0 bit of entropy**

2) Case of collecting at random intervals within a given collection interval

Given collection interval	Estimated entropy
~ 9 ms	0 bit of entropy
10 ~ 99 ms	3 bits of entropy
100 ms ~	6 bits of entropy

Ex) When the given collection interval is 20 ms, collect at random intervals within 20 ms.

- ✓ The number of the values made out of the third : 10
- ✓ We consider that there is no randomness in the first and second z. (Conservative perspective more than CMVP IG)
- ✓ The variability of the zzz values is similar to having 10. ➔ **3 bits of entropy**

Theoretical Analysis - GetTickCount

1. Analyze the structure and characteristics of each noise source.

- GetTickCount is **the number of milliseconds that have elapsed since the system was started**, up to 49.7 days.
- Sample Size : **4 bytes**
- Collecting function : **GetTickCount()**
- The resolution of GetTickCount() : typically **the range of 10 ms to 16 ms**
  - That is, if GetTickCount() is called after a time in the range of 10 ms to 16 ms, GetTickCount is increased by that time.

- The Most variable Byte : LSB(1 byte)
  - **The heuristic analysis is focused on LSB(1 byte).**

Theoretical Analysis - GetTickCount

2. Analyze heuristically the entropy of each noise source.

- Assume that GetTickCount is increased by 16 after every 16ms. (Fix the resolution)
  - The heuristic analysis in CMVP IG is **conservatively used** to estimate entropy.

1) **Case of collecting at regular intervals**
   - The number of the LSB(1 byte) values : **1**
   - Estimated entropy : **0 bit of entropy**

2) **Case of collecting at random intervals within a given collection interval**

Given collection interval	Estimated entropy
~ 31 ms	0 bit of entropy
32 ~ 63 ms	1 bit of entropy
64 ~ 127 ms	2 bits of entropy
128 ~ 255 ms	3 bits of entropy
256 ms ~	4 bits of entropy

Ex) When the given collection interval is 160 ms, collect at random intervals within 160 ms.

- ✓ The number of the LSB(1 byte) values : 10
- ✓ Estimated entropy : **3 bit of entropy**

Theoretical Analysis - GetTickCount

2. Analyze heuristically the entropy of each noise source.

- Assume that GetTickCount is increased by that time after a random time in the range of 10 ms to 16 ms.
  - The heuristic analysis in CMVP IG is **conservatively used** to estimate entropy.

1) **Case of collecting at regular intervals**
   - The number of the LSB(1 byte) values : **7**
   - Estimated entropy : 2 **bits of entropy**

2) **Case of collecting at random intervals within a given collection interval**
   - We assume that the entropy of GetTickCount, which is collected at random intervals within a given collection interval, will be estimated higher than the estimated entropy in the previous slide.

Experimental Analysis - GetSystemTime, GetTickCount

1. Collecting the noise source.

- **Extract the most variable 1 byte among sample and Collect.**
  - Most entropy estimation test can calculate entropy for samples with the size of 8 bits or less.
  - When selecting the most variable 1 byte position, use the Monobit(Frequency) test and Poker-8 test.

- **Collection Options**
  1. Collection interval(ms) : 10, 20, 60, 100, 200, 500
  2. Whether collecting at random intervals within a given collection interval : T/F
  3. The number of samples collecting(byte) : 2000, 5000, 10000, 20000, 50000

- Select and extract the most variable 1 byte every time it is collected according to each collection option.

Experimental Analysis - GetSystemTime, GetTickCount

2. Entropy estimation method

- **Use three algorithms of Min-entropy estimate algorithm of TTA standard.**
  - The Most Common Value Estimate Algorithm
  - The Collision Estimate Algorithm
  - The Compression Estimate Algorithm

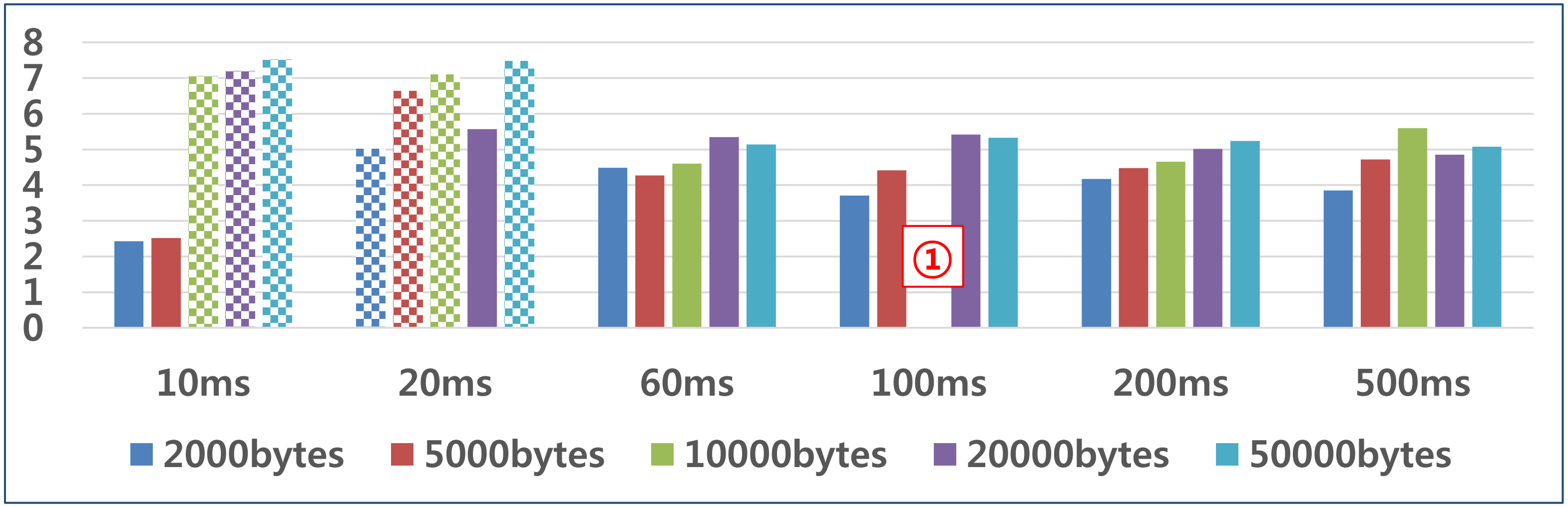
- **The minimum of Min-entropy estimated by each of three algorithms**

  = Min-entropy of the noise source

- Since the collision estimate algorithm and the compression estimate algorithms use the numerical method, these can operate abnormally.
- So **Min-entropy of the noise source is obtained except for the result of the abnormal operation of the algorithm.**
  - The result of the abnormal algorithm is represented by this pattern in the graph.

Experimental Analysis - GetSystemTime

1. **Collect at random interval within given collection interval and Estimate entropy of the collected noise source.**
   - pattern : **Min-entropy of the noise source obtained except for the result of the abnormal operation of the algorithm.**



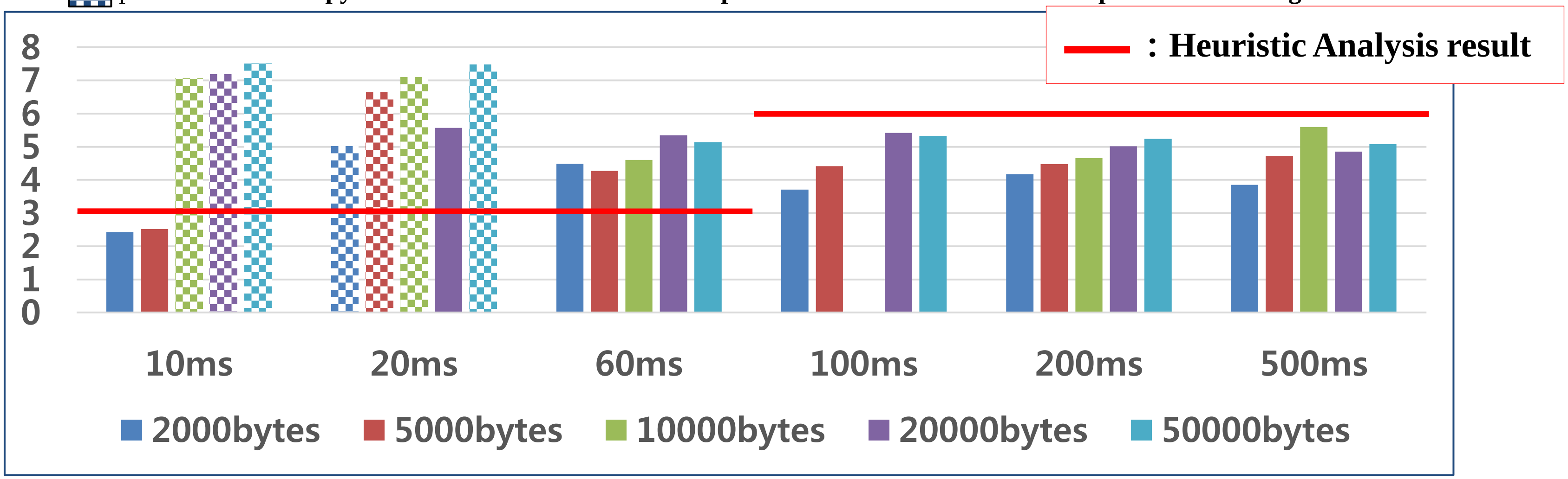
- 100 ms & 10,000 bytes (①) : 0 bit of entropy
  - The most variable 1 byte position selected : **Date-related position**
  - The most variable 1 byte position selected from other results except for : **Milliseconds-related position (LSB 1 btye)**

GetSystemTime is appropriate to extract and collect the 1 byte related to milliseconds in the sample, especially 1 byte located in the LSB(1 byte) position of milliseconds.

Experimental Analysis - GetSystemTime

1. **Collect at random interval within given collection interval and Estimate entropy of the collected noise source.**

- pattern : **Min-entropy of the noise source obtained except for the result of the abnormal operation of the algorithm.**



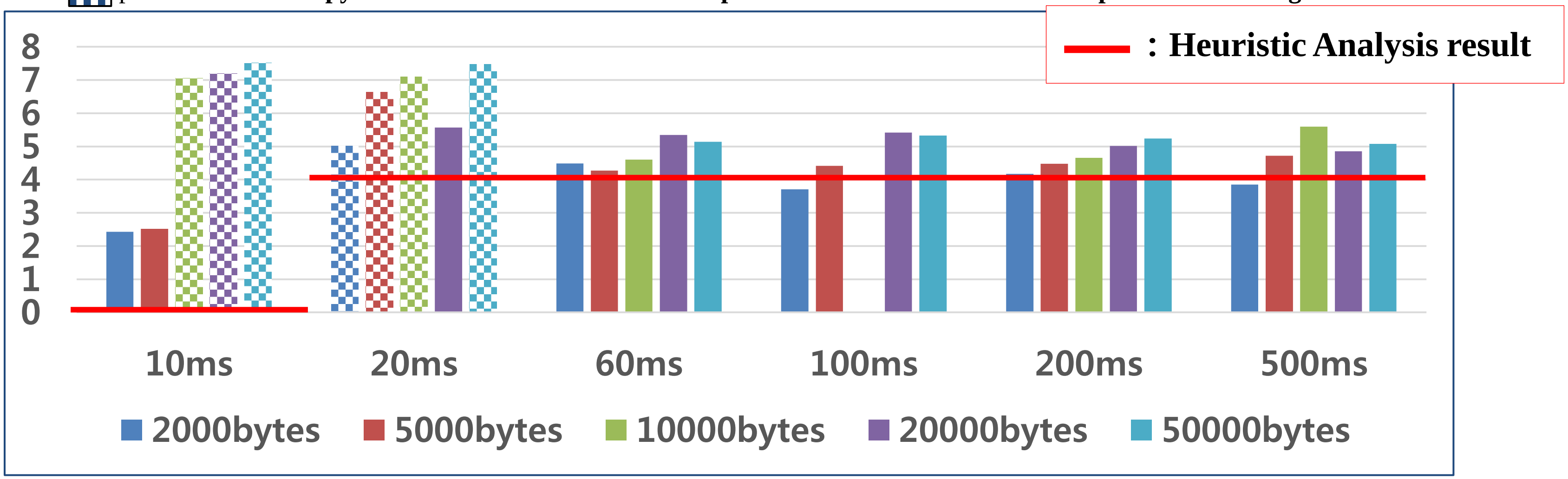
- Experimental results do not follow the heuristic analysis results.
  - Heuristic Analysis
    - Focus on the number of the zzz values. (zzz = milliseconds)
    - **Full-entropy : 10 bits of entropy**
  - Experiment
    - Focus on the most variable 1 byte.
    - **Full-entropy : 8 bits of entropy**

Do heuristic analysis of GetSystemTime with full-entropy of 8 bits

Experimental Analysis - GetSystemTime

1. **Collect at random interval within given collection interval and Estimate entropy of the collected noise source.**

   - (checkered pattern) pattern : **Min-entropy of the noise source obtained except for the result of the abnormal operation of the algorithm.**



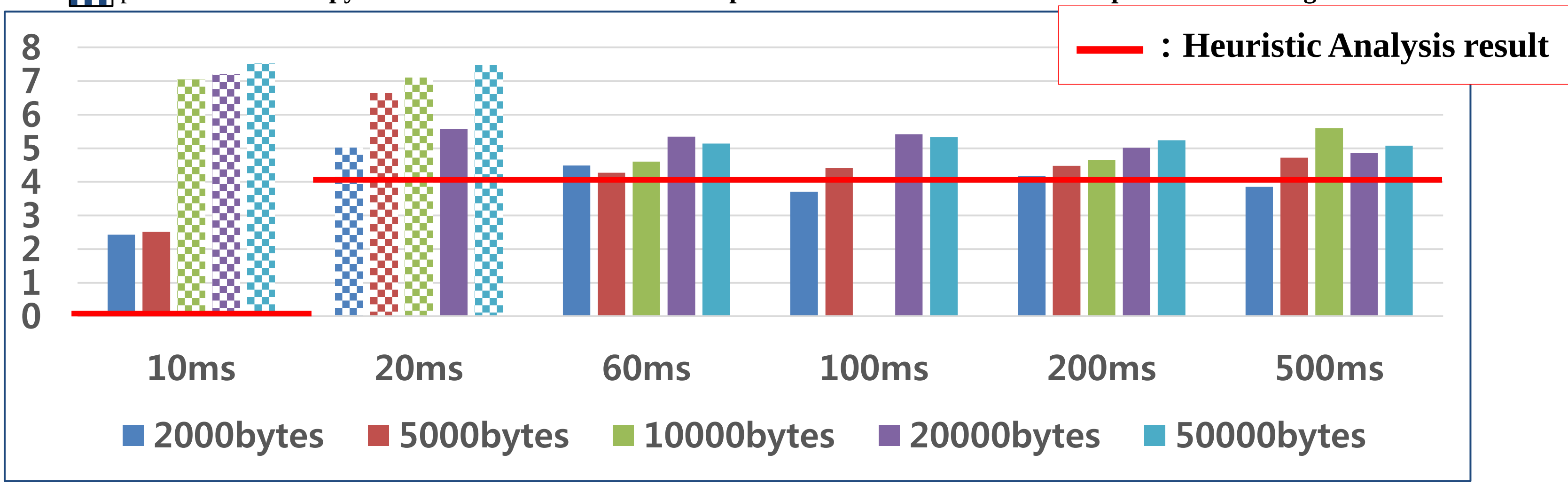
- Heuristic analysis of GetSystemTime with full-entropy of 8 bits
  - Focus on LSB(1 byte) of zzz. (zzz = milliseconds)

Given collection interval	Estimated entropy
~ 15 ms	0 bit of entropy
16 ms ~	4 bits of entropy

Experimental Analysis - GetSystemTime

1. **Collect at random interval within given collection interval and Estimate entropy of the collected noise source.**

- pattern : **Min-entropy of the noise source obtained except for the result of the abnormal operation of the algorithm.**



- Experimental results almost follow heuristic analysis results.

Min-entropy of the collected GetSystemTime

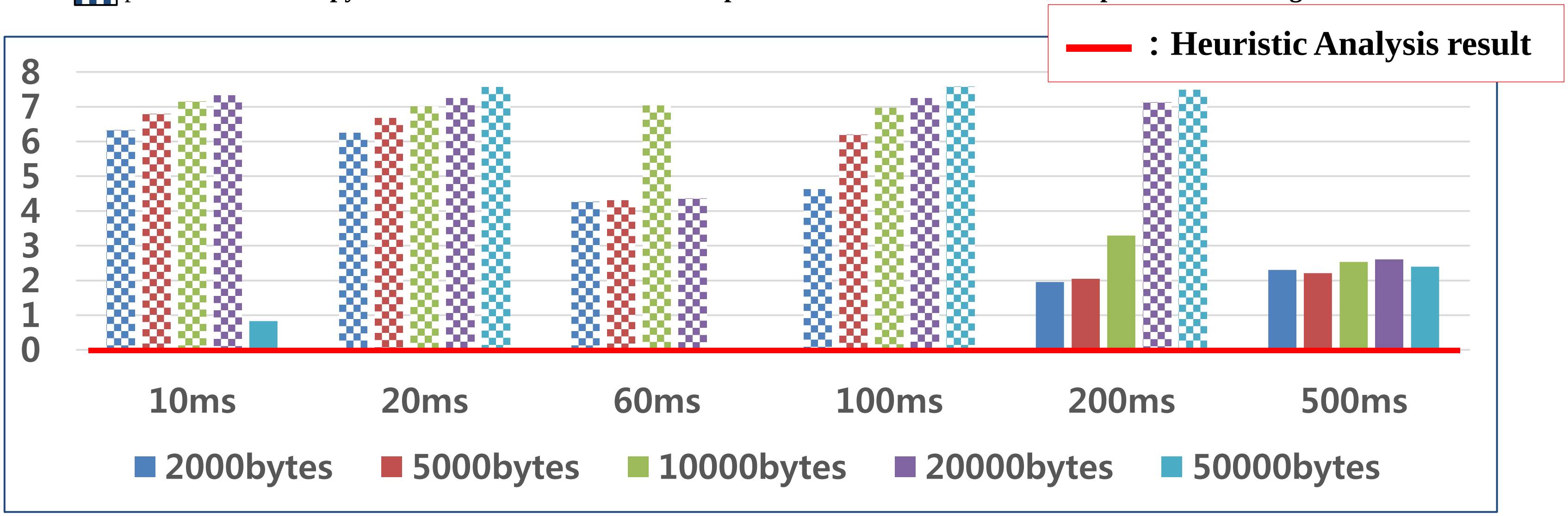
= min (Min-entropy estimated by the estimation algorithm,

Min-entropy estimated by heuristic analysis)

Experimental Analysis - GetSystemTime

2. **Collect at regular intervals and Estimate entropy of the collected noise source.**

- pattern : **Min-entropy of the noise source obtained except for the result of the abnormal operation of the algorithm.**



- Min-entropy of the collected GetSystemTime

  = min (Min-entropy estimated by the estimation algorithm,

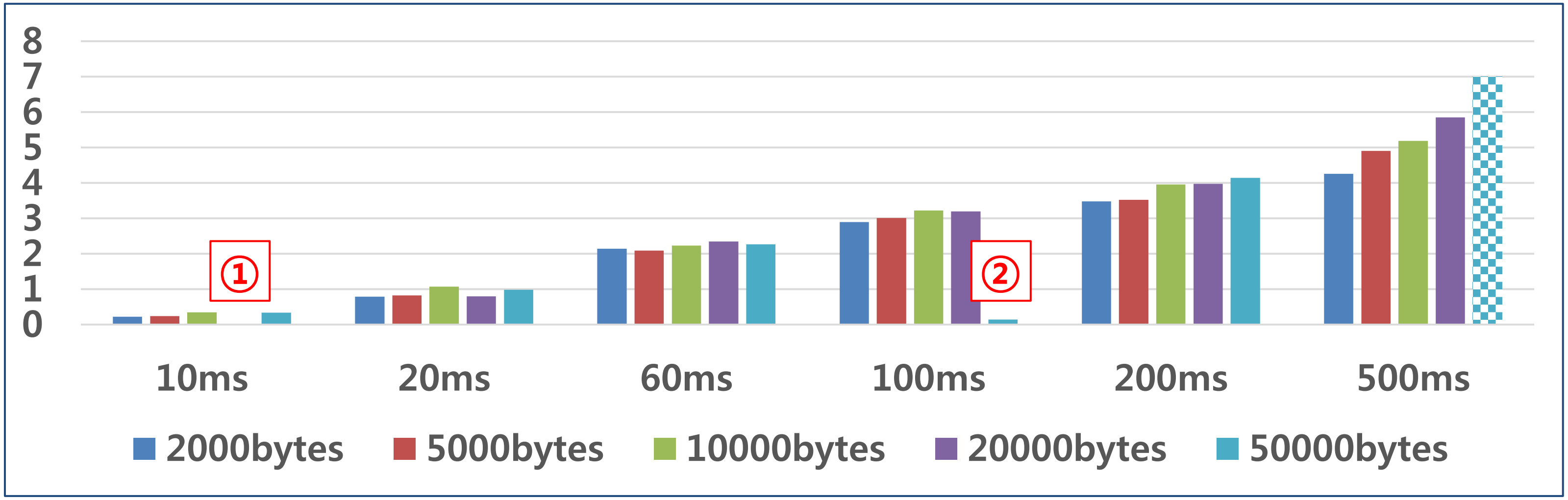
  Min-entropy estimated by heuristic analysis)

  = 0 bit of entropy

It is desirable to collect GetSystemTime at a random interval within a given collection interval rather than a regular collection interval.

Experimental Analysis - GetTickCount

1. **Collect at random interval within given collection interval and Estimate entropy of the collected noise source.**
   - pattern : **Min-entropy of the noise source obtained except for the result of the abnormal operation of the algorithm.**

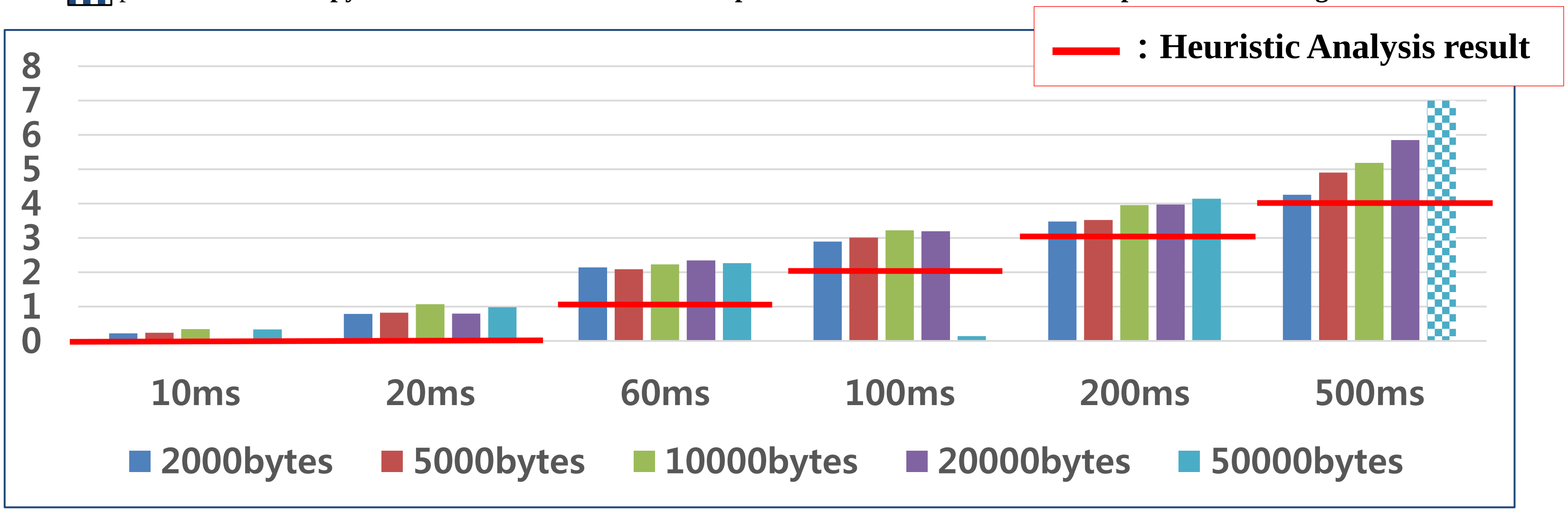


- 10 ms & 20,000 bytes (①) or 100 ms & 50,000 bytes (②) : 0 bit of entropy
  - The most variable 1 byte position selected : **Non-LSB(1 byte)**
  - The most variable 1 byte position selected from other results except for : **LSB(1 btye)**

GetTickCount is appropriate to extract and collect the LSB(1 byte) of the sample.

Experimental Analysis - GetTickCount

1. **Collect at random interval within given collection interval** and Estimate entropy of the collected noise source.
   - pattern : **Min-entropy of the noise source obtained except for the result of the abnormal operation of the algorithm.**



- Experimental results almost follow heuristic analysis results.

Min-entropy of the collected GetTickCount

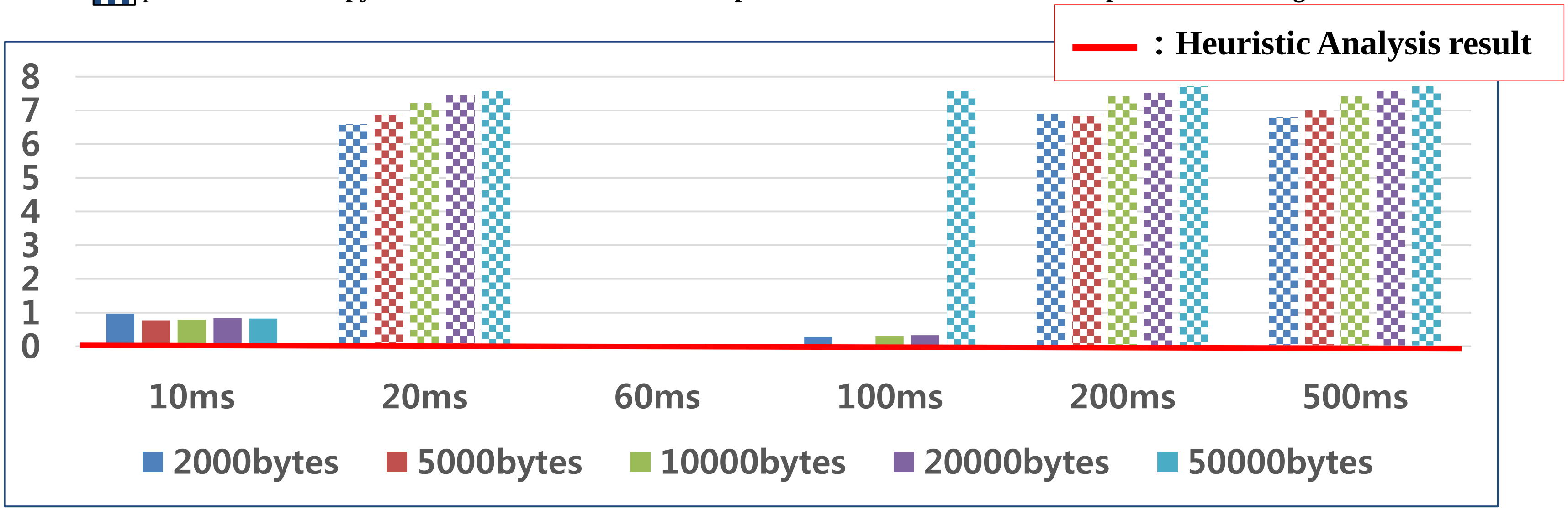
= min (Min-entropy estimated by the estimation algorithm,

Min-entropy estimated by heuristic analysis)

Experimental Analysis - GetTickCount

2. **Collect at regular intervals and Estimate entropy of the collected noise source.**

- pattern : **Min-entropy of the noise source obtained except for the result of the abnormal operation of the algorithm.**



- Min-entropy of the collected GetTickCount

  = min (Min-entropy estimated by the estimation algorithm,

  Min-entropy estimated by heuristic analysis)

  = 0 bit of entropy

It is desirable to collect GetTickCount at a random interval within a given collection interval rather than a regular collection interval.

Thus…

- **Heuristic Analysis Results**

	GetSystemTime (Full-entropy : 10 bits)	GetSystemTime (Full-entropy : 8 bits)	GetTickCount (Full-entropy : 8 bits)
The most variable 1 byte	milliseconds	LSB(1 byte) of milliseconds	LSB(1 byte)

	Given Collection Interval	Estimated Entropy	Given Collection Interval	Estimated Entropy	Given Collection Interval	Estimated Entropy
Case of collecting at random intervals within a given collection interval	~ 9 ms	0 bit of entropy	~ 15 ms	0 bit of entropy	~ 31 ms	0 bit of entropy
	10 ~ 99 ms	3 bits of entropy	16 ms ~	4 bits of entropy	32 ~ 63 ms	1 bit of entropy
	100 ms ~	6 bits of entropy			64 ~ 127 ms	2 bits of entropy
					128 ~ 255 ms	3 bits of entropy
					256 ms ~	4 bits of entropy

Case of collecting at regular intervals	0 bit of entropy	0 bit of entropy	0 bit of entropy

- **Comparison of experimental results and heuristic analysis results**
  - It is desirable to select the most variable 1 byte position considering the characteristics of the noise source.
  - It is desirable to collect at random intervals within a given collection interval.
  - In order to accurately evaluate the entropy, the heuristic analysis results and the experimental results should be complementary. Min-entropy of the noise source = min(experimental results, heuristic analysis results)

Reference

- NIST, Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program, August, 2016.
- M.S. Turan et al. Recommendation for the Entropy Sources Used for Random Bit Generation, (Second DRAFT)NIST Special Publication 800-90B, Jan. 2016.
- Wolfgang Killmann and Werner Schindler, “A Proposal for : Functionality Classes and evaluation methodology for true(physical) random number generators”, BSI AIS.31, September, 2001.
- MSDN, https://msdn.microsoft.com/
- KCMVP, http://www.nis.go.kr/AF/1_7_3_3/list.do

Thank you!