The Current Status and Entropy Estimation Methodology in KCMVP

in ICMC16

2016.5.18

National Security Research Institute
SeogChung Seo, SangWoon Jang

Kookmin University
Yongjin Yeom

KCMVP?

KCMVP

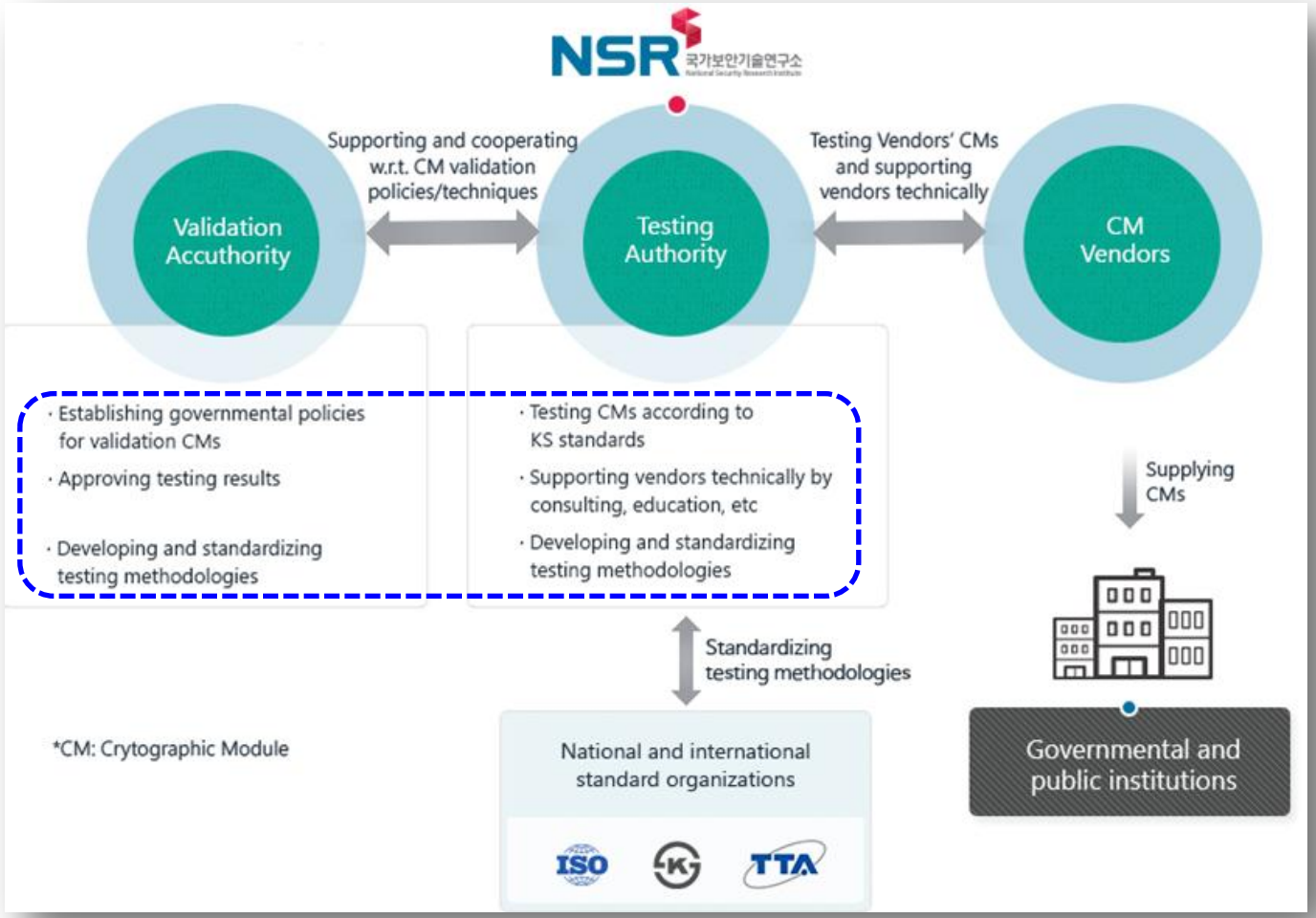
Korea Cryptographic Module Validation Program

- Validating security and implementation conformance of a CM for the protection of sensitive information in governmental/public institutions
- CAVP is conducted in KCMVP process
- Since 2005

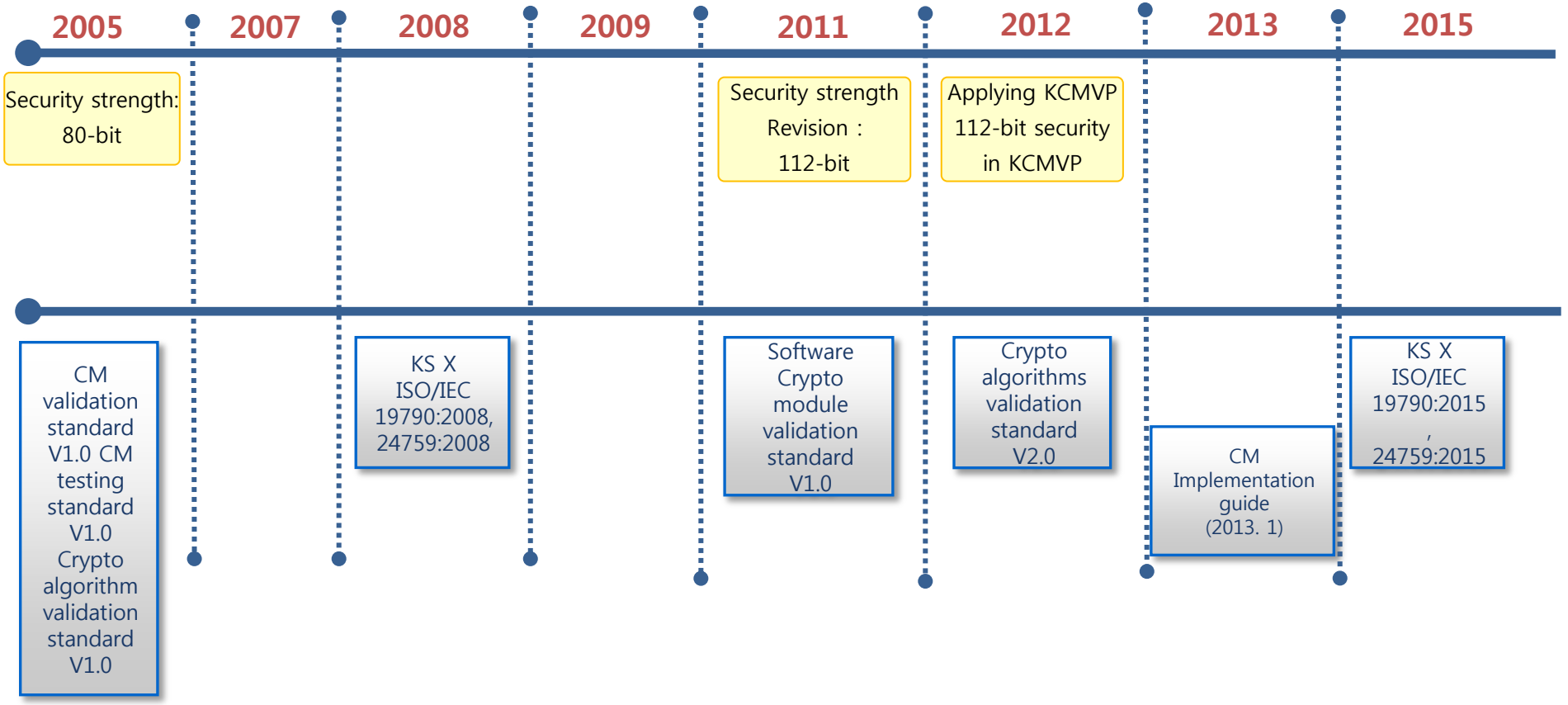
Standard

- Security Requirements : KS X ISO/IEC 19790:2015
- Test Requirements : KS X ISO/IEC 24759:2015
- Approved Algorithms : ISO/IEC, KS, TTA

KCMVP?

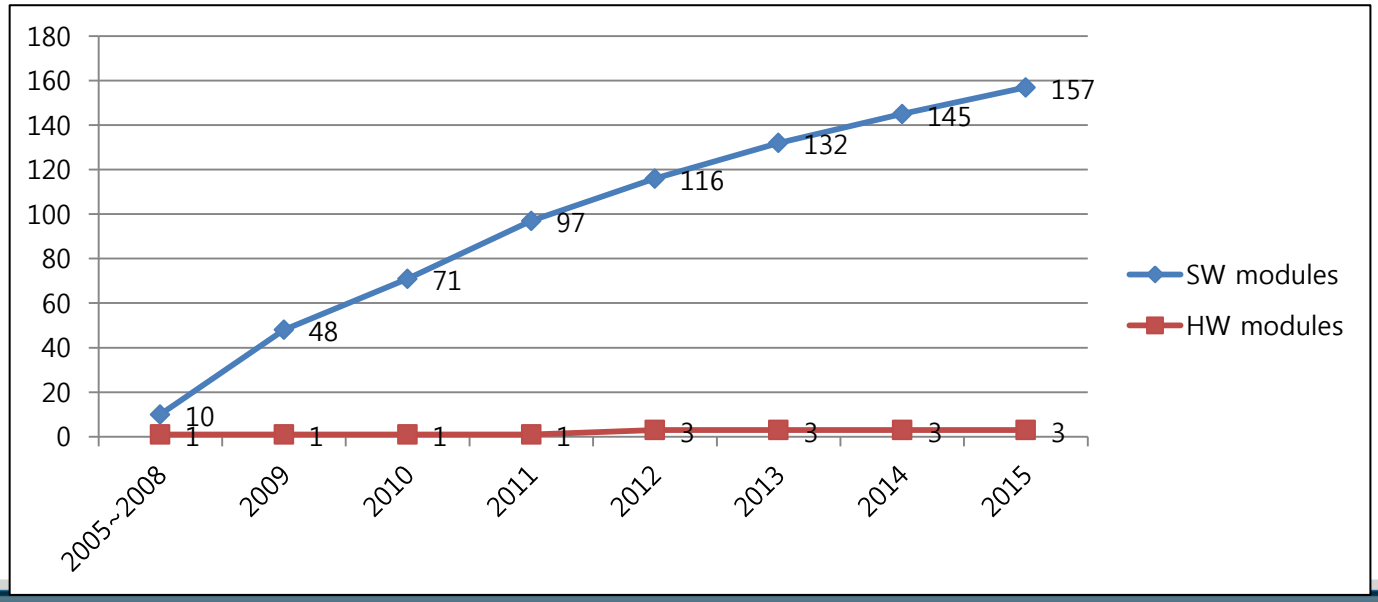

NSR
국가보안기술연구소
Validation Accuthority
Supporting and cooperating w.r.t. CM validation policies/techniques
Testing Authority
Testing Vendors' CMs and supporting vendors technically
CM Vendors
· Establishing governmental policies for validation CMs
· Approving testing results
· Developing and standardizing testing methodologies
· Testing CMs according to KS standards
· Supporting vendors technically by consulting, education, etc
· Developing and standardizing testing methodologies
Supplying CMs
Standardizing testing methodologies
*CM: Crytographic Module
National and international standard organizations
ISO
TTA
Governmental and public institutions

History of KCMVP



Current validation statistics in KCMVP

- 160 modules are validated
  - **SW modules: 157, HW modules: 3**
    - Library: 100(User library >> Kernel library, C > Java > C++)
    - Crypto Applications: 57
- Operational environments
  - **Windows > Linux/Unix > Java > Mobile(android, iOS)**



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, ECDH
Public key encryption		RSAES
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.

KCMVP Test Process

CAVP
with KCAVS

Conformance test

CM Specification	- Understanding overall structure of CM - Checking approved mode & algorithms & CSP
Finite State Model	Consistency btw FSM and execution of CM
Roles, Services and Authentication	Supports concurrent users or not?
CM Ports and Interfaces	- Checking consistency btw documents and code - Checking unidentified information flows - Checking underlying cryptographic algorithms
Operational Environment	Testing CM on every operational environments
Key Management	- Conducting **Entropy analysis** - Checking key life management generation, store, input/output, zeroisation, etc - Checking underlying DRBGs
Self Tests	Conducting negative & positive selftests
Design Assurance	Checking development environments
Mitigation of Other Attacks	Conducting **Source code security analysis**

- Document review
- Source code analysis
- Func&Operational test

CAVP in KCMVP

- **CAVP test tool**
  - **KCAVS 4.1(revised at 2016)**
- **Most frequently used approved algorithms in KCMVP**
  - **Symmetric-key > Hash > MAC > RBG > Public key encryption > Digital signature > KE**
- **CAVP duration**
  - **Depends on types of algorithms**
    - KE > Public key algorithms > RBG > MAC/Hash > Symmetric
  - **Depends on vendor's knowledge on crypto and development**
  - **Need to develop CAVP test applications**

CAVP in KCMVP

- **Vendor's difficulties in CAVP**
  - **Effort to develop CAVP test applications → get better with sample codes**
  - **Lack of understanding CAVP process → get better with education**
    - DRBG test for RBG
    - MCT test for symmetric key
    - Public key algorithms (encryption, digital signature, KE), etc
  - **Format error → get handled with file I/O utilities**
  - **Algorithmic defects → easily find faults with log comparison**
    - Key/parameter generation in public key algorithms, CMAC/CCM, DRBG, etc
- **Our plan to develop methods for speeding CAVP up**
  - **Providing CAVP sample codes and utilities for easy file I/O**
    - Consider Endian, word size, operational environments, language
    - Consider algorithmic characteristics and CM's interfaces
    - Logging functions for tracking the algorithmic failure

Difficulties in Entropy Assessment

- It is infeasible to achieve perfect randomness in ideal coin-toss (unpredictable, unbiased, independent)
- It is required to prove (or show the evidence of) sufficient closeness to ideal randomness
- It is hard to develop criteria for lower bound of entropy estimations

Particularly for software modules,

- We have to justify whether the module collects sufficient entropy from the operating environment (not within the module)
- Most noise sources used in a software module are non-physical sources provided by OS

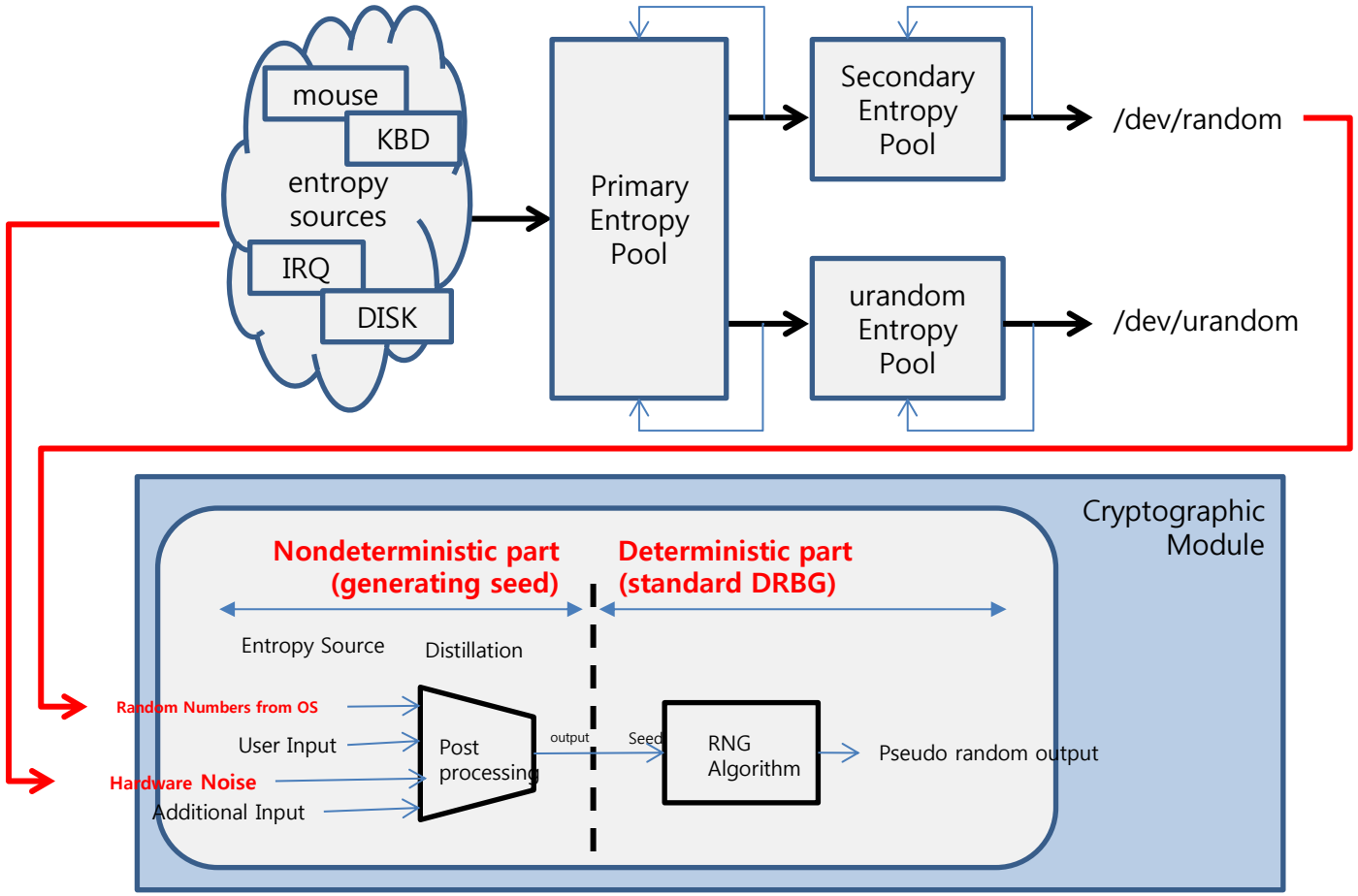
Referenced documents

- (SP 800-22) Statistical Randomness Tests
  - Suitable for evaluating final output of RNGs
  - Not applicable to digitized entropy sources (without post-processing/conditioning)
- (BSI AIS.31) Test for TRNGs
  - Evaluation criteria for TRNG(Physical RNG)s
  - Suitable for detecting physical failures (easy to pass)
- (SP 800-90B) Test for entropy sources (draft)
  - Conservative entropy assessment for noise sources

➔ Hard to adopt them to assess entropy in software modules

RNG in a software module

- Software module and its operating environment



Entropy sources in SW module

- Collecting entropy from the operating environment
- Not completely determined when the module is tested in KCMVP Lab.

The problems are…

- KCMVP has to finish the entropy assessment without knowing that the exact operating environment (only knowing the OS)
- It is hard to collect sufficient data for entropy estimation

Examples of entropy sources

- System functions such as GetCurrentThreadID(), GetCurrentProcessID(), and GetCursorPos()
- Output of RNG in OS: /dev/random, CryptGenRandom(), etc.

Estimation of entropy rate

- When collecting entropy from the physical noise source,
  we assume that each sample is harvested independently
- Then entropy rate (entropy per bit) can be regarded as constant

➔ If we estimate the amount of entropy in a single sample conservatively),
total entropy is expected to increase linearly

➔ Entropy estimations in 800-90B focus on the lower bound of entropy per sample

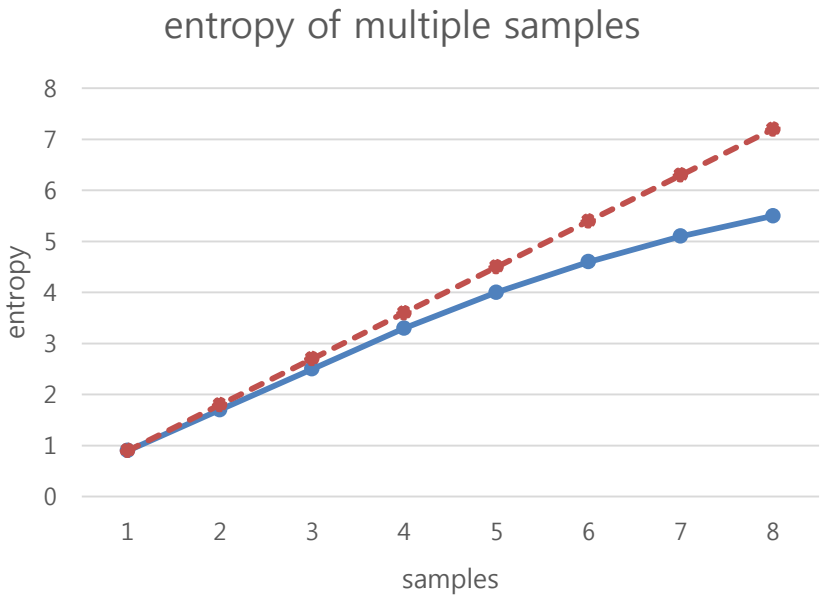
- However, samples are dependent (particularly in SW noise)
- Total entropy is not proportional to the number of samples
  (observed by the experiments)

Entropy estimation for multiple samples

- **Motivation**
  - CMVP requires to collect hundreds of bit of entropy for DRBG
  - How many times do we have to collect samples iteratively for sufficient entropy?
  - It is desirable to estimate the lower bound of entropy for multiple samples collected by repeated harvests

- **Observation**
  - Experimental results show that entropy does not increase linearly

➔ How can we find a lower bound?



Min-entropy of random variables

- **Notations**
  - Min-entropy of a random variable X : $H_\infty(X)$
  - Min-entropy of (X, Y) for the joint distribution : $H_\infty(X, Y)$
  - Product distribution $X \times Y$: $\mathrm{P}_{X \times Y}(X, Y) = P(X)P(Y)$

- **When we collect entropy from a single source repeatedly,**
  - Sequence of samplings : $X_1, X_2, \ldots, X_n$
  - Joint distribution (X_1, X_2) : distribution for double size sampling
  - Upper bound : $H_\infty(X_1, X_2) \leq H_\infty(X_1) + H_\infty(X_2) = 2H_\infty(X_1)$

➔ Can we have a lower bound for $H_\infty(X_1, X_2)$?

Lower bound Theorem for min-entropy

- **THEOREM for Lower bound**

  Let X_1 and X_2 be random variables for the repeated sampling model whose relative independence $RI(X_1, X_2)$ is bounded by ϵ ($RI(X_1, X_2) \leq \epsilon$). Then

$$H_\infty(X_1, X_2) \geq H_\infty(X_1) + H_\infty(X_2) - \epsilon \ln e \ .$$

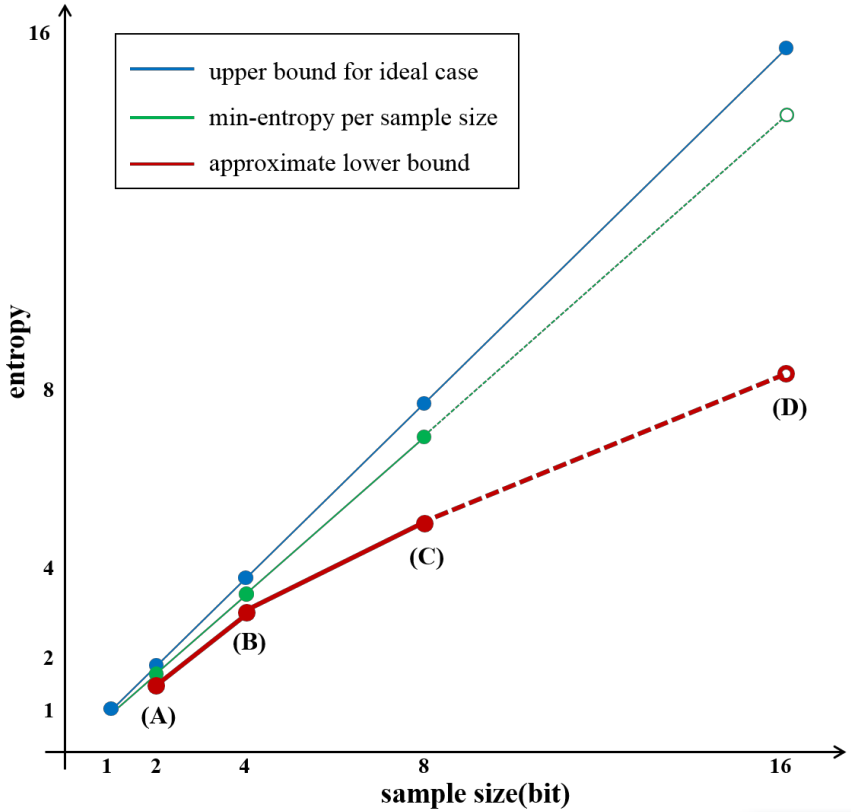
- **Remark**
  - Relative independence $RI(X, Y)$ is the rate of dependency defined as the smallest $\epsilon > 0$ s.t.

$$P_{X\times Y}(X, Y)\, exp(-\epsilon) \leq P_{(X,Y)}(X, Y) \leq P_{X\times Y}(X, Y)\, exp(\epsilon)$$

  - THEOREM says that if samples are collected repeatedly, the min-entropy of two adjacent samples has a lower bound determined by their relative independence
  - Applying THEOREM repeatedly, we have a lower bound for multiple samples

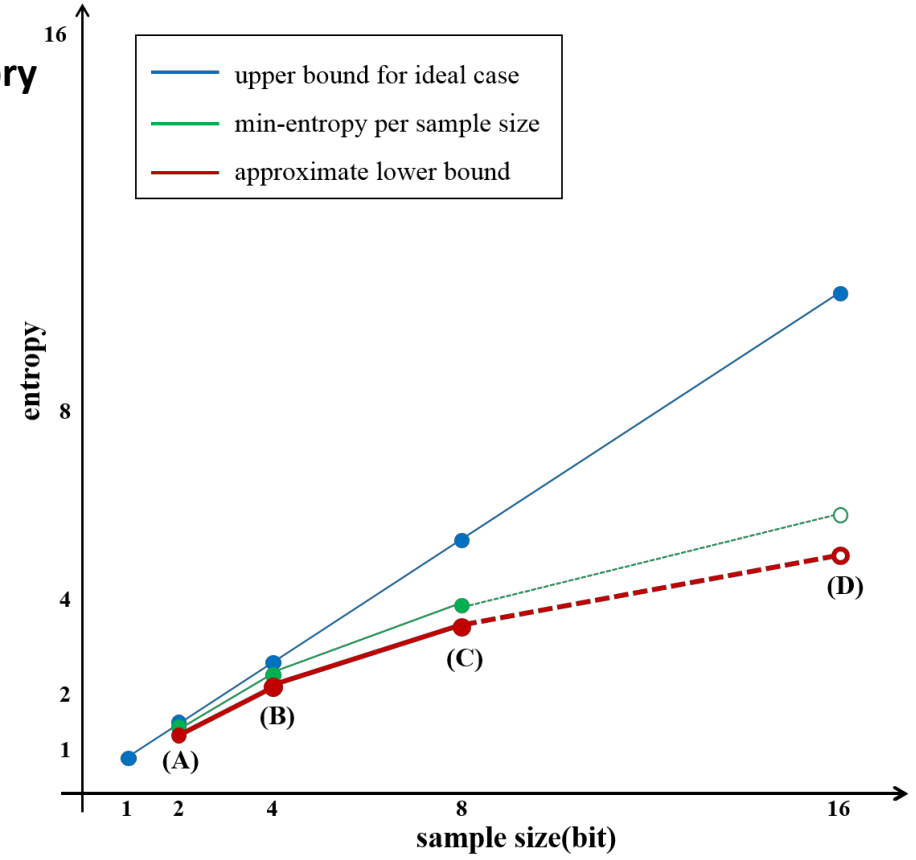
Example : min-entropy estimation

- Example: IID source
  - HW RNG: Quantum RNG (Quantis-USB by IDQ)
  - Sample size: 1 bit
  - 800-90B estimations for various sample sizes increase almost linearly
  - The lower bounds given by THEOREM seem to be very conservative



Example : min-entropy estimation

- Example: non-IID source
  - Entropy source: GPU (NVIDIA GTX780)
  - Rationale: Race conditions on shared memory
  - Sample size: 1 bit
  - 800-90B estimations (green curve) does not increase linearly (showing dependency)
  - The lower bounds given by THEOREM seem to be tight



Summary of entropy estimation

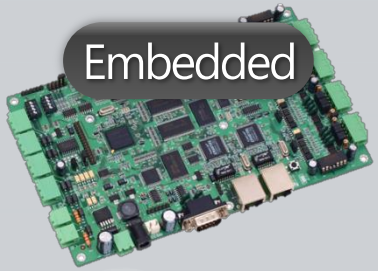
- In KCMVP,
  - Require minimum 112-bit entropy on raw noise data
  - Most of approved cryptographic modules are software modules
  - Entropy sources lies outside of the modules, mostly provided by OSes
  - It is recommended to collect entropy as many sources as possible

- Several approaches for estimating entropy for software modules
  - Lower bound estimations are important to ensure that the module collect sufficient entropy for DRBG
  - Because of the limitations of data size, we develop variants of existing statistical tests
  - Mutual information between consecutive samples are considered to estimate entropy

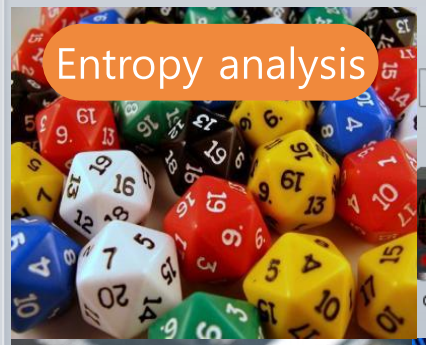
Future in KCMVP?

Developing new testing methods

New ICT CM Validation



Enhancing validation of core component techniques



Q & A

Q&A