

W H I T E W O O D

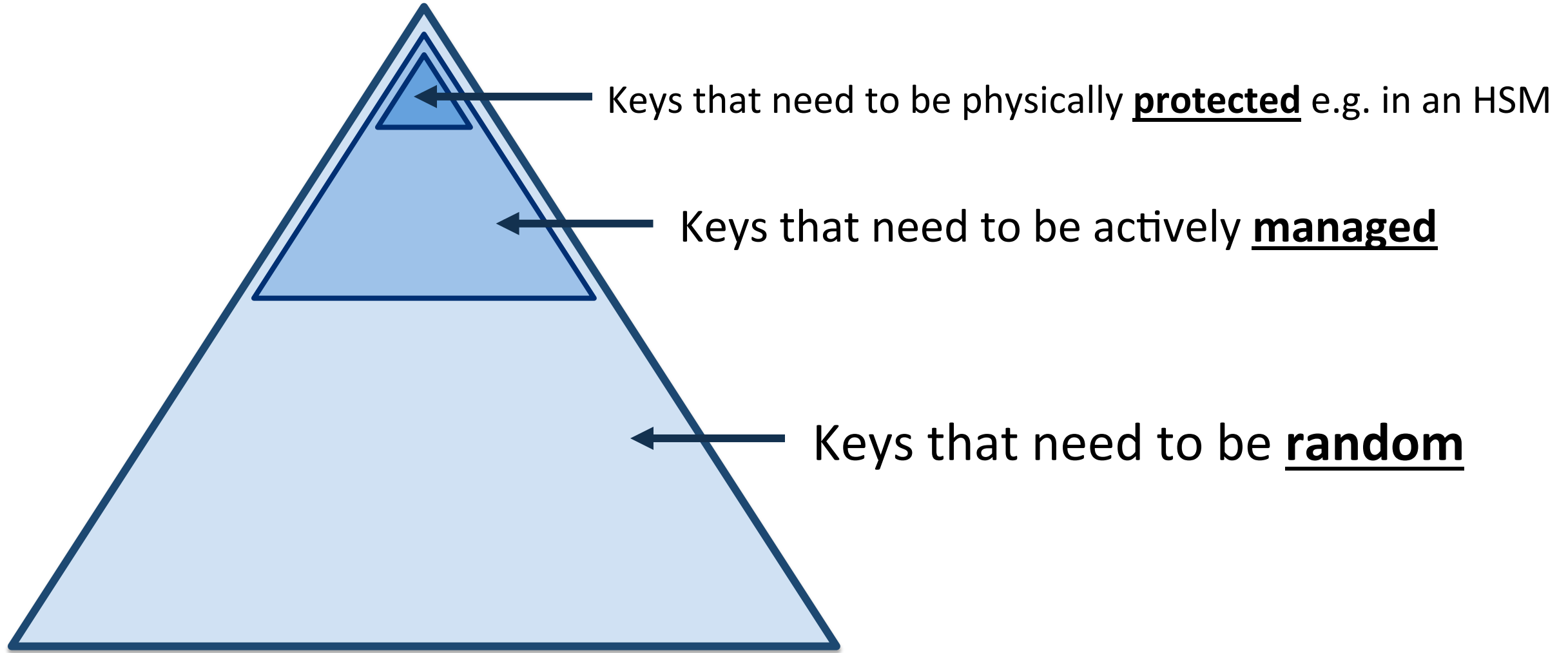
Entropy – A Case of Supply and Demand

ICMC 2017 – Washington DC

Richard Moulds – General Manager, Whitewood

May 19th 2017

Keys to the kingdom



All crypto security starts with random numbers

2537013740134601340136408134013481349010
3948109834679019183409813467134876901313
7069710394761034758347590478019348163486
0183409183476890134681756907629084691158
6928628761053465013874650134051435014365
1346501734605134051345018374601340134786
0134786014380157840896710347861034786501
3785613405134081345183745651756136450613

Crypto security assumptions rely on keys being random, when patterns emerge (or are engineered) keys get more predictable

Anything less than true randomness is a risk

But, there's a problem

We need more and more randomness

But, it's getting harder to be truly random

- More and more crypto in use
- Longer and longer keys
- Increased key management scrutiny
- Tougher compliance
- Quantum threat...

- Abstraction, containers and VMs
- Hosted and cloud environments
- Headless systems, no 'users'
- Snap shots and replication
- Low power IoT devices

Hidden vulnerabilities - backdoors of choice

NIST NIST Time | NIST Home | About NIST | Contact Us | A-Z Site Index Search

Information Technology Laboratory

About ITL ▾ Publications Topic/Subject Areas ▾ Products/Services ▾ News/Multimedia Programs/Projects

NIST Removes Cryptography Algorithm from Random Number Generator Recommendations

From NIST Tech Beat: April 21, 2014

The revised document retains three of the four previously available options for generating pseudorandom bits needed to create secure cryptographic keys for the Random Bit Generator. NIST recommends that current users continue to use the existing algorithms as long as they are possible.

Schneier on Security

Blog Newsletter Books Essays News Speaking

Random Number Bug in Debian Linux

This is a [big deal](#):

The Register
Biting the hand that feeds IT

Windows random number generator is so not random

All too predictable

Not-So-Random Numbers in Virtualized Linux and the Whirlwind RNG

ZDNet SEARCH Q SECURITY DATA CENTERS VIDEOS CXO WINDOWS 10

Security flaw leaves Android Bitcoin wallets vulnerable to theft

Bitcoin wallets generated on Android are thought to be suffering from a random number generation weakness.

By Liam Timp | August 12, 2013 -- 10:21 GMT (02:21 PDT) | Topic: Security

ZDNet Q VIDEOS

The seven most dangerous attack techniques: A SANS Institute rundown

From ransomware to weak random number generators, the RSA Conference explores the worst threats and how to stop them.

By Stephanie Condon for Between the Lines | February 15, 2017 -- 20:52 GMT (12:52 PST) | Topic: Security

Thomas Ristenpart, Michael Swift
Computer Sciences
University of Wisconsin-Madison
{ristenpart,swift}@cs.wisc.edu

Randomness Exposed – An Attack on Hosted Virtual Machines

Christopher J. Thompson, Ian J. De Silva, Marie D. Manner,
Michael T. Foley, and Paul E. Baxter

info security Latest
STRATEGY | INSIGHT | TECHNOLOGY

Lord David Blunkett Urges Orgs to take Cyber Highway to Better Security

20 FEB 2013 MAGAZINE FEATURE

The Dark Side of Cryptography: Kleptography in Black-Box Implementations

the morning paper

selected by Adrian

Mining your Ps and Qs: Detection of Widespread Weak Keys in Network Devices

SEPTEMBER 16, 2015

threat post CATEGORIES FEATURED PODCASTS VIDEOS

SIEMENS PATCHES INSUFFICIENT ENTROPY VULNERABILITY IN ICS SYSTEMS

by Tom Spring December 22, 2016, 12:28 pm

WHITEWOOD

Basic requirements for randomness

Uniformity: As many 1s as 0s, on average

Independence: Each bit uncorrelated with all previous - statistical tests

- Diehard(er), NIST SP800-22 STS, TestU01 etc.

These are necessary, but are **not** sufficient:

- π passes these tests –

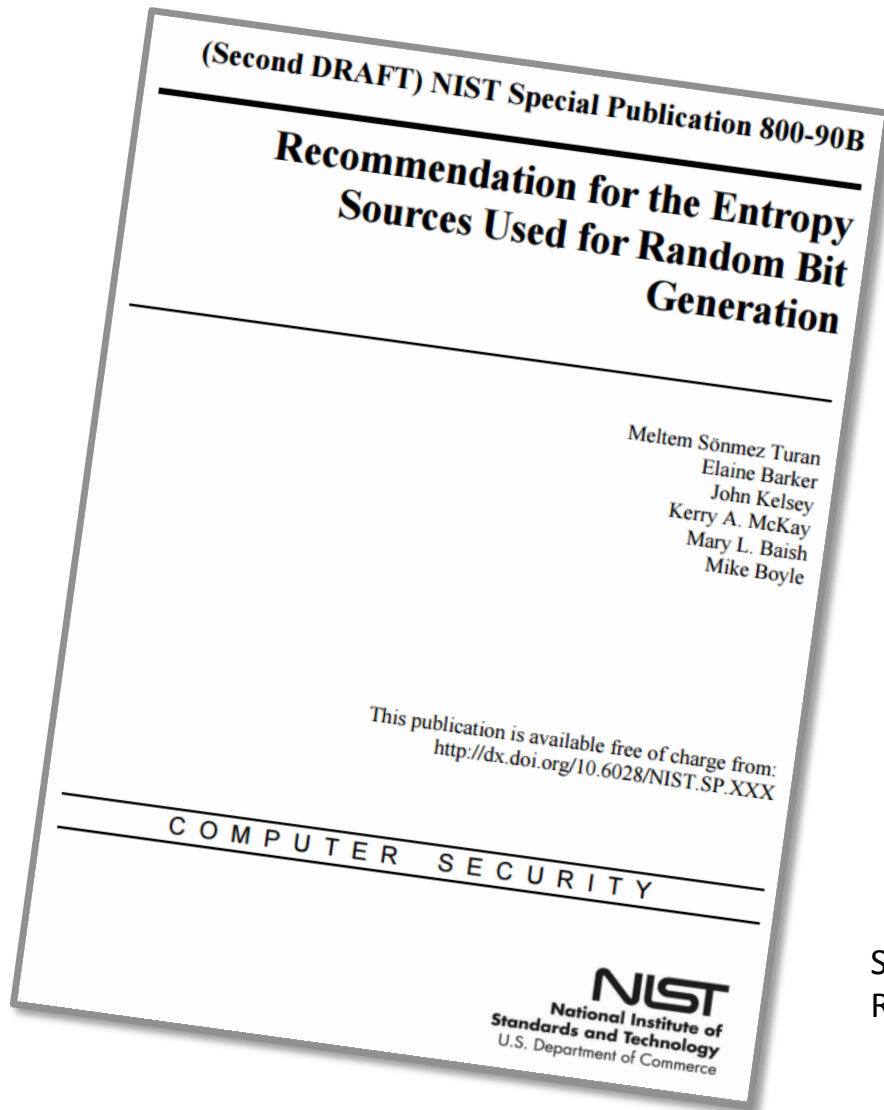
3.141592653589793238462643383279502884197169399375896

For cryptography, we also need -

- Unpredictability – forward and backward protection
- Imperturbability
- Secrecy
- Reliability

Proving true randomness requires knowledge of the source of randomness, not just statistical analysis of the output

Finally we have a standard (nearly)

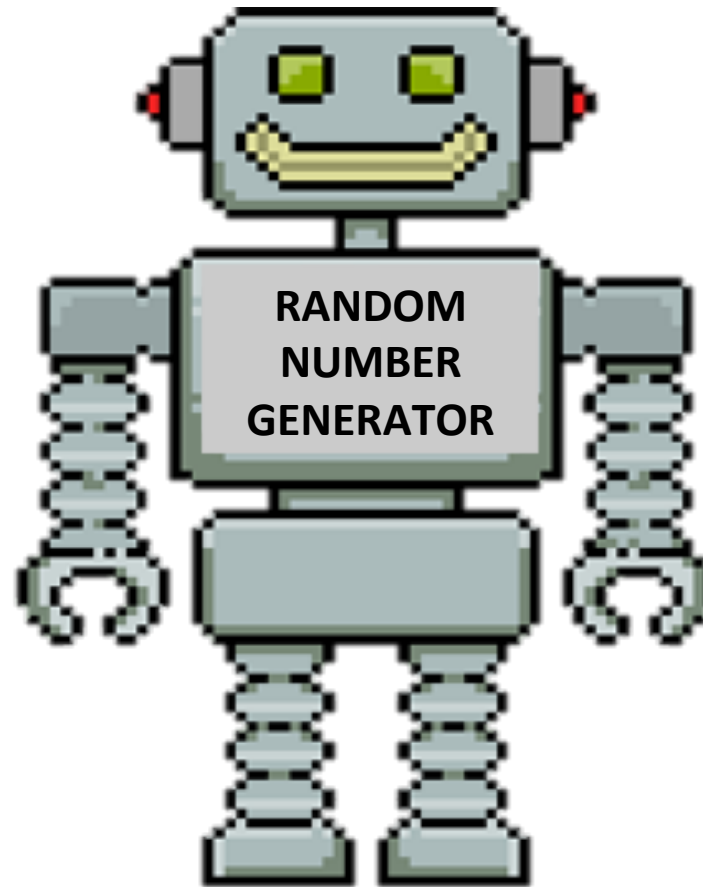


“Specifying an entropy source is a complicated matter. This is partly due to confusion in the meaning of entropy, and partly due to the fact that, while other parts of an RBG design are strictly algorithmic, entropy sources depend on physical processes that may vary from one instance of a source to another”.

Source – Recommendation for the Entropy Sources Used for Random Bit Generation (SP800-90B 2nd draft) – NIST January 2016

Why so complicated?

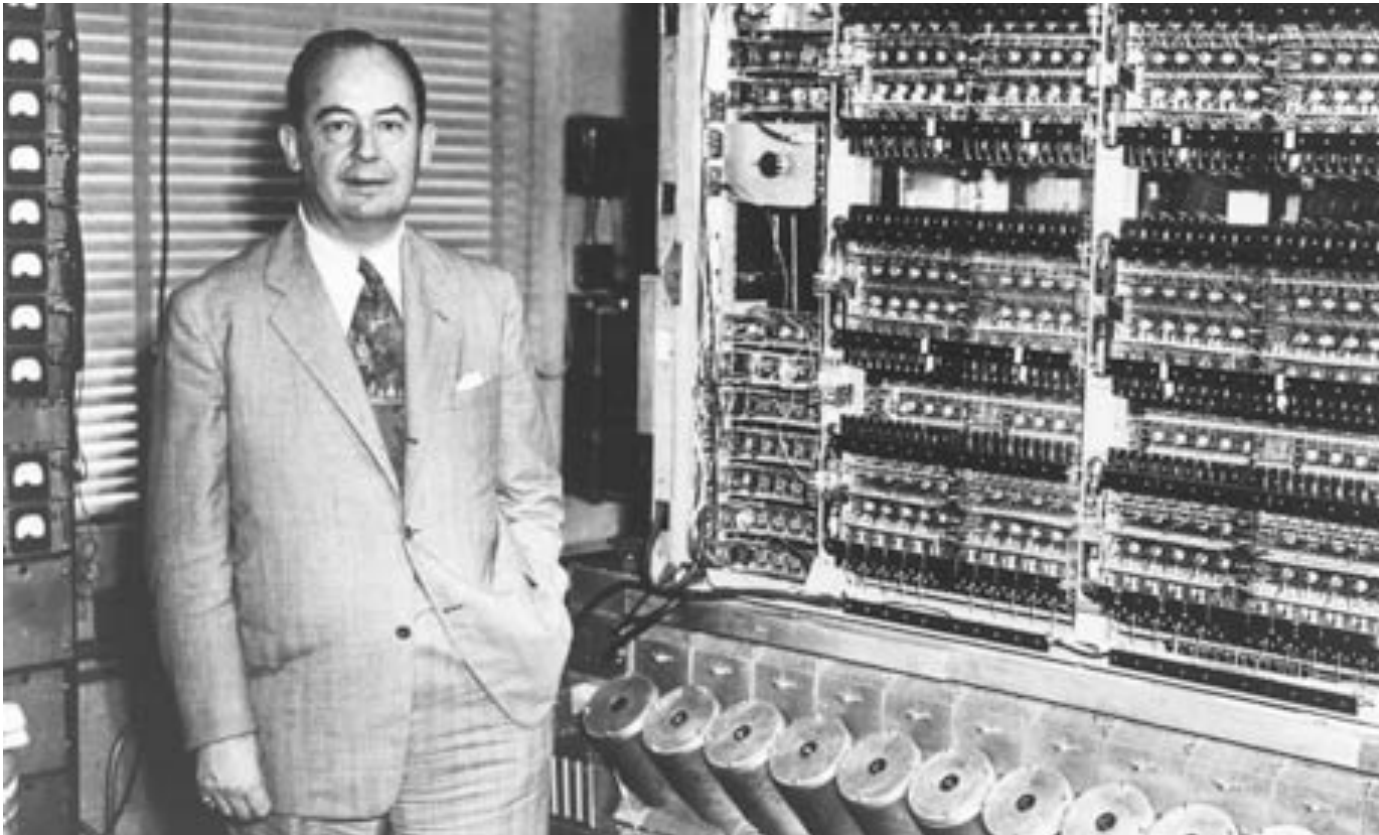
**Most random
numbers
come from
the Operating
System**



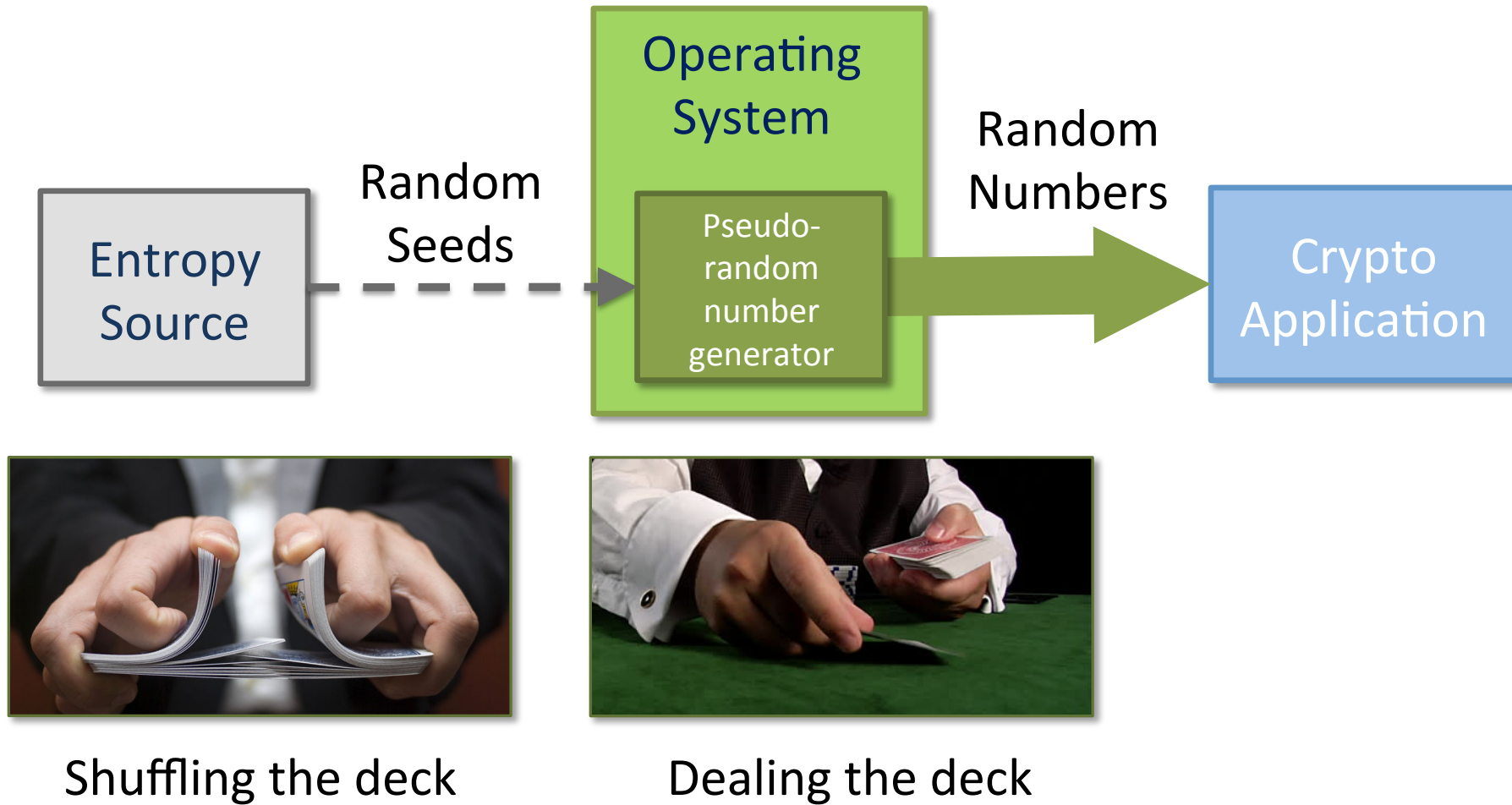
**But
software
doesn't act
randomly**

Entropy - a long standing issue

“Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin.” (J. von Neumann, 1951)



Pseudo-random numbers – an oxymoron?



PRNG assumptions

- PRNGs are well defined and easily validated (SP800-90A)
- In principle need very minimal seeding

	SHA-1	SHA-224 and SHA- 512/224	SHA-256 and SHA- 512/256	SHA-384	SHA-512
Maximum entropy input length (<i>max_length</i>)	$\leq 2^{35}$ bits				
Seed length (<i>seedlen</i>) for Hash_DRBG	440	440	440	888	888
Maximum personalization string length (<i>max_personalization_string_length</i>)	$\leq 2^{35}$ bits				
Maximum additional input length (<i>max_additional_input_length</i>)	$< 2^{35}$ bits				
<i>max_number_of_bits_per_request</i>	$\leq 2^{19}$ bits				
Number of requests between reseeds (<i>reseed_interval</i>)	$\leq 2^{48}$				

Security assumptions

- Sufficient amount of entropy
- Sufficient quality of entropy
- Secrecy of seed values
- Secrecy of internal state
- Independence of internal state across multiple instances

Lack of confidence in assumptions should force up the reseeding rate

Entropy supply and demand - CISO's reality



Payment terminals



User devices



Private datacenter



Private clouds



Public clouds



Google Cloud Platform



Network infrastructure

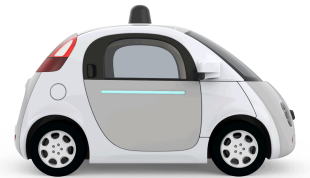


CISO's Confidence Level?

- Quantity
- Quality
- Secrecy
- independence

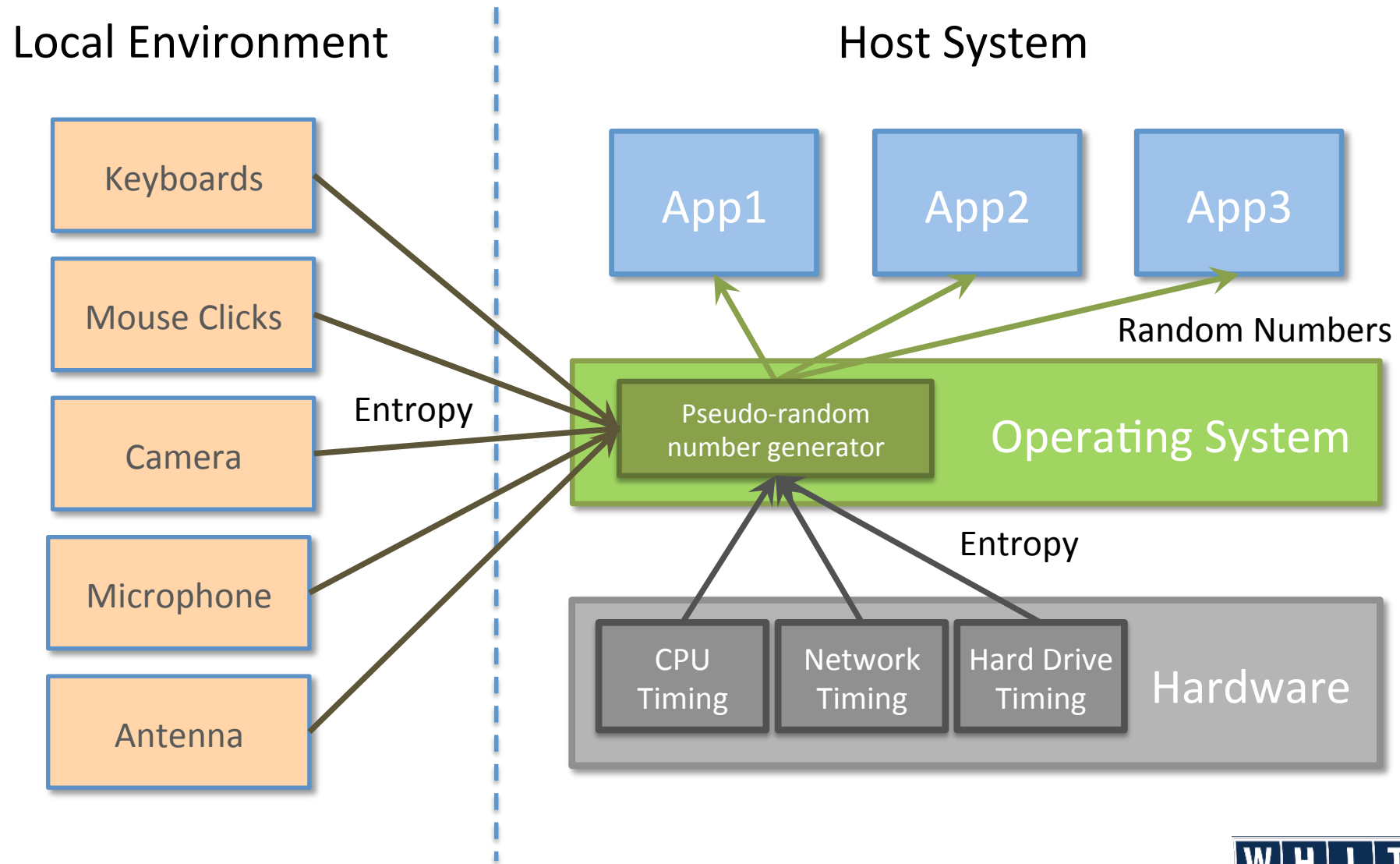


IoT devices

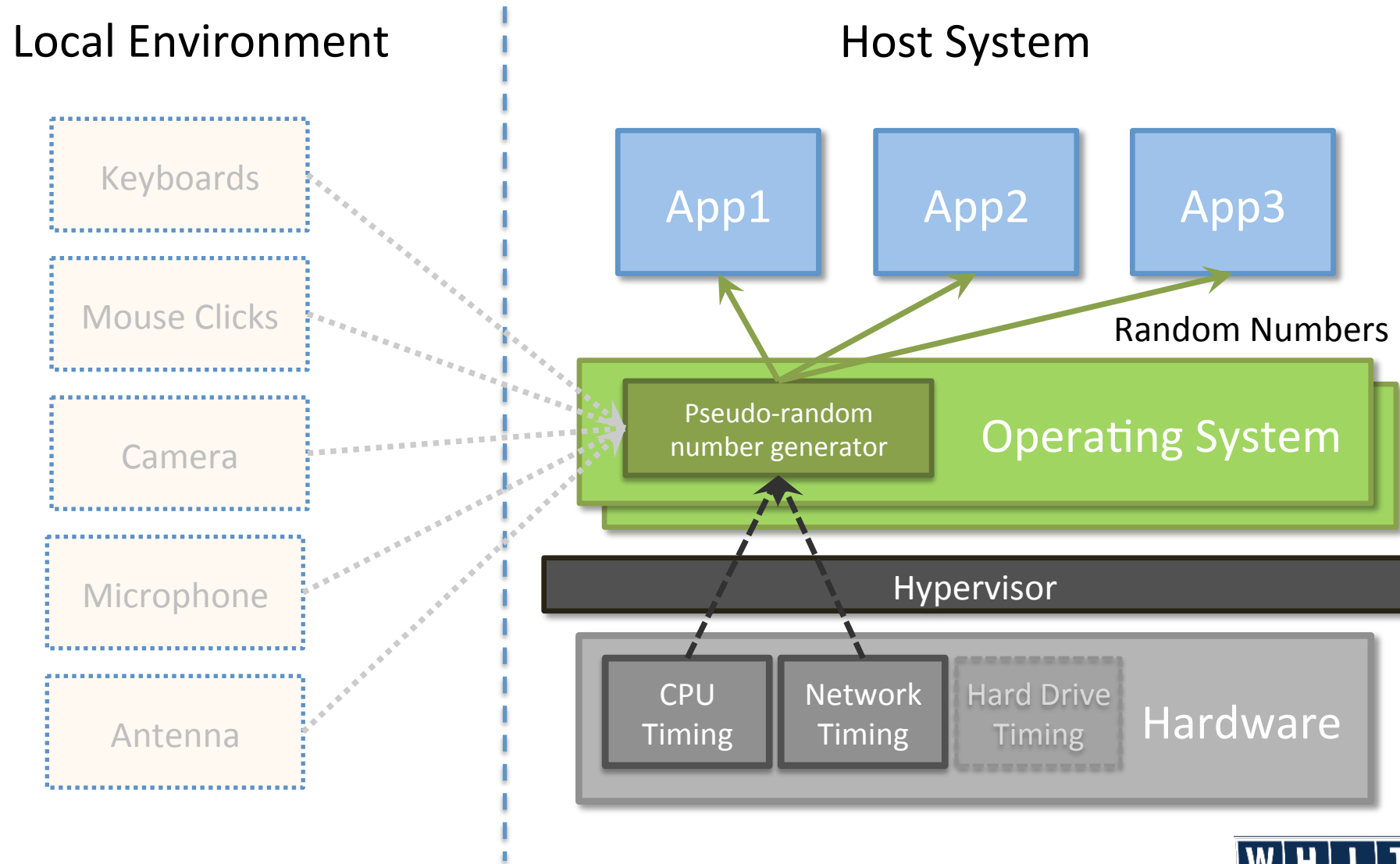


WHITEWOOD

Where does entropy come from?



But in a virtual world...



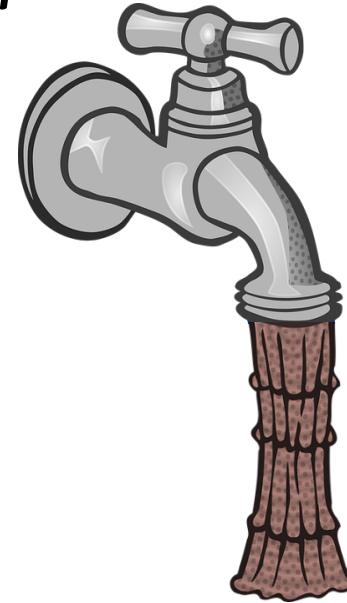
Random number generators in Linux

dev/random



Delivers random numbers only if sufficient entropy has been captured - otherwise it stops

dev/urandom



Delivers random numbers irrespective of how much entropy has been captured

Interrupt derived entropy in Linux

Kernel IRQ handler adds data from interrupts into the Entropy Pool

Cycle Count & Kernel Timer
(4 bytes)

IRQ
(4 bytes)

Instruction Pointer
(8 bytes)

Cycles	Kernel	IRQ	Instruction Pointer
123975895488	4294893898	14	18446744071578900000
123977123888	4294893898	14	18446744071578900000
123979445304	4294893898	14	18446744071578900000
123983781984	4294893899	14	18446744071578900000
123985083096	4294893899	14	18446744071578900000
123986825584	4294893899	14	18446744071578900000
123987250920	4294893899	14	18446744071578900000

Thanks to Adam Everspaugh - <http://pages.cs.wisc.edu/~ace/>

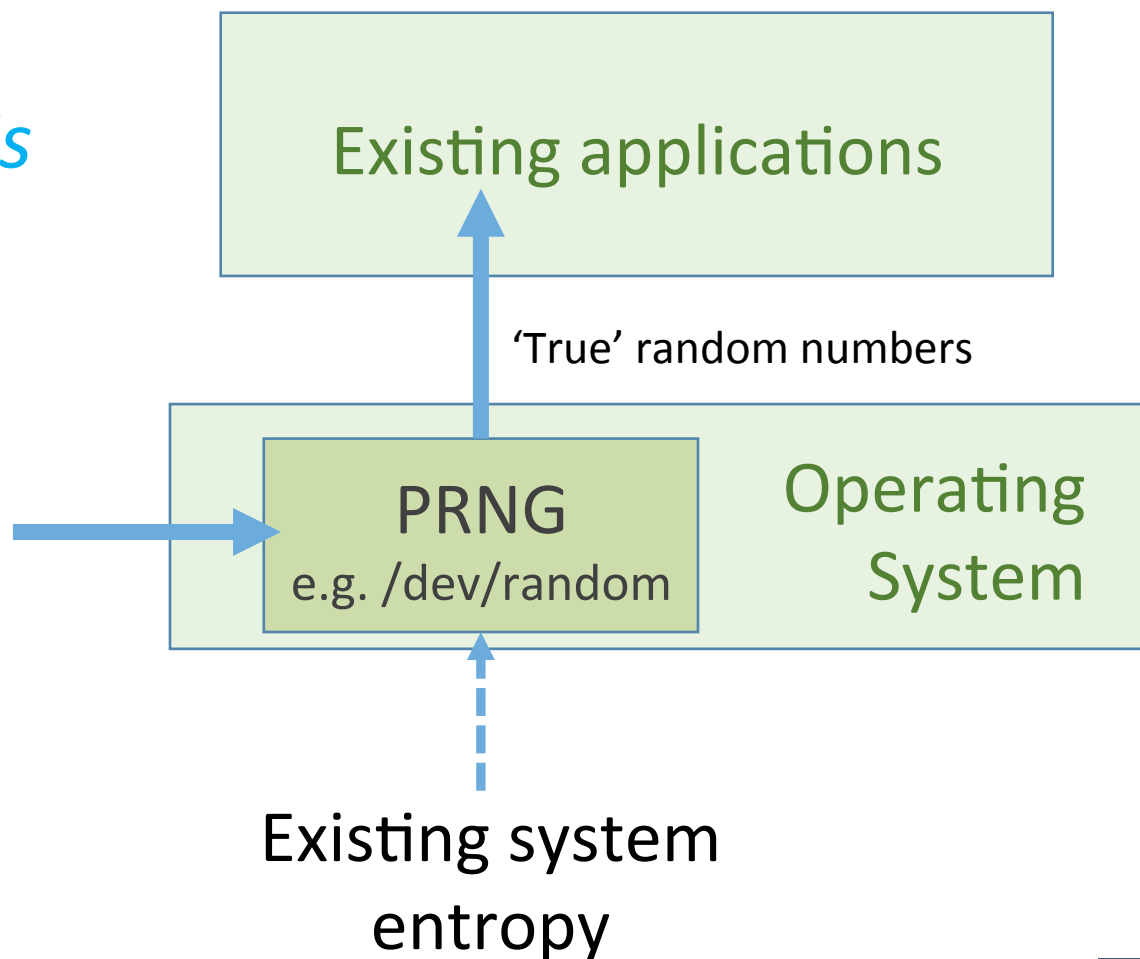
WHITEWOOD

Enhancing system randomness

Goal: generate true random numbers from a PRNG

*Good news - Entropy is
always additive*

Supplementary
entropy source(s)



Supplementary sources of entropy

Three general approaches to improve entropy:

1. Software daemons to more efficiently extract entropy from existing signals and interrupts
 - State changes - HAVEGED (www.issihosts.com/haveged/)
 - Timing Jitter – CPU Jitter RNG (www.chronox.de/jent.html)
 - Microphones and cameras – ‘noise’ sources (www.vanheusden.com/aed/ and www.lavarnd.org/)
2. Local hardware based entropy sources
 - Embedded CPU feature - e.g. Intel RdRand
 - Plug-in devices - USB sticks, HSMs, PCI cards, etc.
3. Network based services for entropy sources and RNGs
 - Random number services (<https://random.org> and <https://beacon.nist.gov/home>)
 - “Entropy as a Service” (<https://getnetrandom.com> and <https://entropy.ubuntu.com/>)

Comparison of supplementary entropy sources

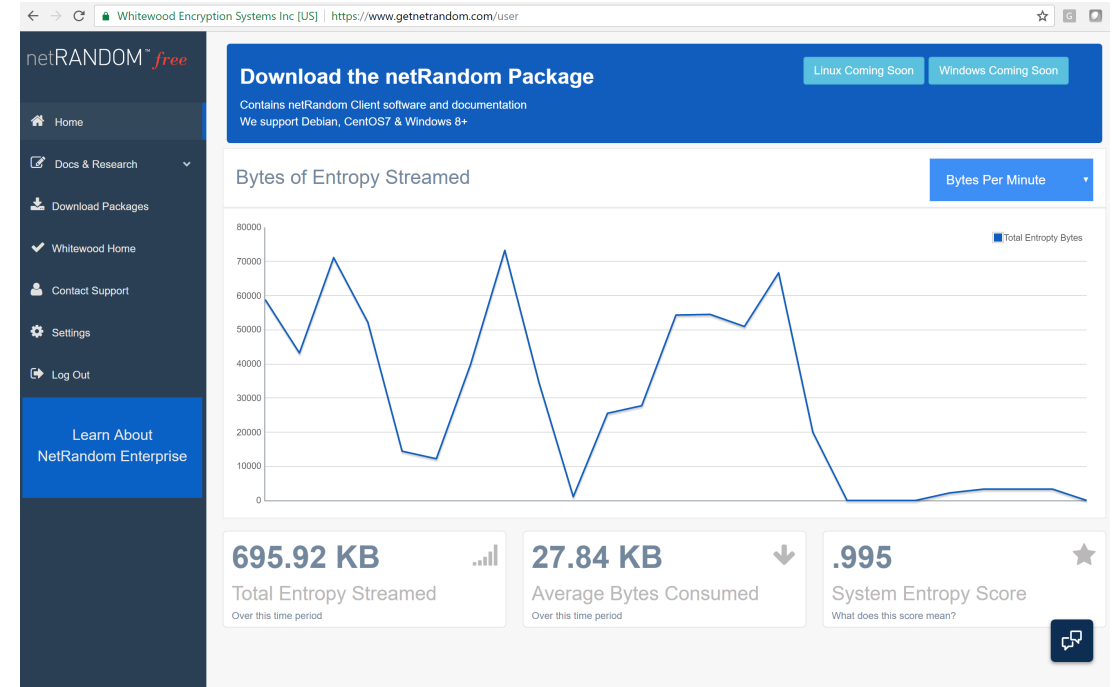
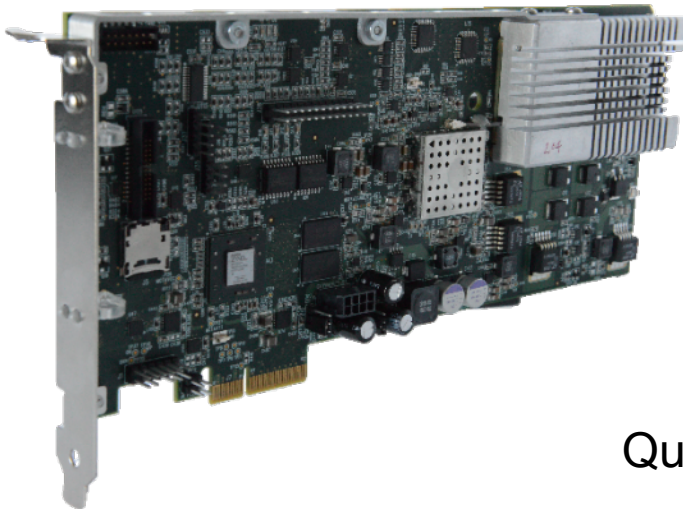
	Entropy Extractors	Embedded Hardware Sources	Retrofit Hardware Sources	Entropy as a Service
Form factor	Software	Platform hardware	Plug-in hardware	Service (public or private)
Advantages	<ul style="list-style-type: none"> • Low cost (free) • Lots of choices • Open source 	<ul style="list-style-type: none"> • Low cost (free?) • Convenient • Speed 	<ul style="list-style-type: none"> • High assurance • Speed 	<ul style="list-style-type: none"> • Consistency • Scale • Assurance • Low cost (free?)
Barriers	<ul style="list-style-type: none"> • Hard to validate • Hard to Manage • Open source 	<ul style="list-style-type: none"> • Platform specific • Hard to validate • Requires configuration 	<ul style="list-style-type: none"> • Inconvenient • Not cloud friendly • Cost 	<ul style="list-style-type: none"> • Immaturity • Complexity (?)

Shameless plug

netRANDOM™

Network Delivered Quantum Entropy
(Entropy as a Service)

- netRandom Free – cloud service - getnetrandom.com
- netRandom Enterprise - private, on-premise entropy servers



EntropyEngine™

Quantum Random Number Generator



Summary

- Random numbers are critical for security but are often poorly understood and managed
- Random number generators are a point of attack and vulnerability – potentially an invisible one
- Modern application environments present entropy challenges – VMs, cloud and IoT
- Testing the quality of entropy sources and random number generators is hard
 - new standards will help (NIST SP 800-90B)
- Supplementary sources of entropy improve security and there are plenty of deployment options
- Random number generation should be a critical component of your key management strategy and datacenter infrastructure planning

Thank you

richard.moulds@whitewoodsecurity.com

Entropy-as-a-Service at www.getnetrandom.com

Demo at www.whitewoodsecurity.com/netrandom-demo

