Uses and applications of randomness

The built of a secure and trustable source of random number generator

About the presenter – Raphael Machado

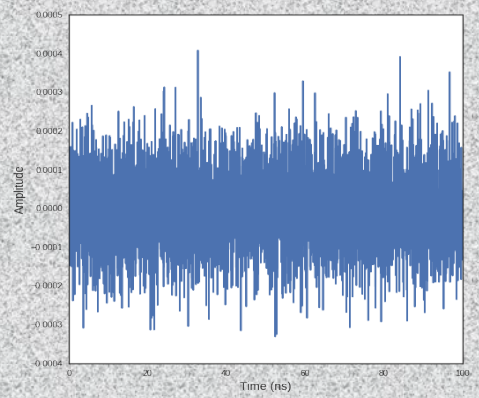
- Researcher at Inmetro (Brazilian National Institute of Metrology, Quality and Technology)
  - Chief of Informatics Laboratory
  - Coordinator of Graduate Course of Metrology and Quality
- Professor at Federal Center for Technological Education (Rio de Janeiro)
  - Chair of Algorithms Design and Analysis
- Leader of the Brazilian Randomness Beacon Project (FAPESP)
- Leader of the SHCDCiber Project (Ministry of Defense)
- Leader of other TCS and InfoSec projects (CNPq, FAPERJ)
- Young Scientist (FAPERJ) and Research Productivity (CNPq) grants

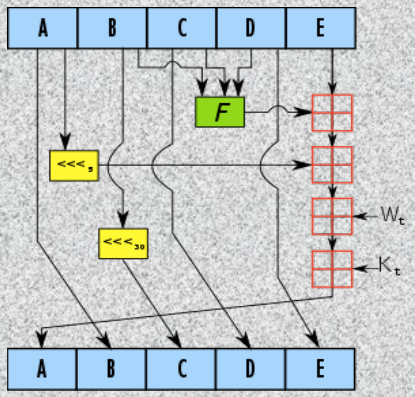
First of all: what is a random event

A mathematical definition:
The output of a probabilistic experiment

The output is...
42

A pratical definition:
Anything that <u>appears</u> to be random





Where do we use randomness

- Games, lottery, cassino, bingo...
- Selection of judges and jurors
- Computer simulations
- Test of hypotheses - randomized controlled trial
- Computer security: key generation, nonces,...

Where do we use randomness

- **Games, lottery, cassino, bingo...**
- Selection of judges and jurors
- Computer simulations
- Test of hypotheses - randomized controlled trial
- Computer security: key generation, nonces,...

Where do we use randomness

- **Games,**
- Selectio
- Comput
- Test of h
- Computer security: key generation, nonces,...

Where do we use randomness

- Games, lottery, cassino, bingo...
- **Selection of judges and jurors**
- Computer simulations
- Test of hypotheses - randomized controlled trial
- Computer security: key generation, nonces,...

Whe

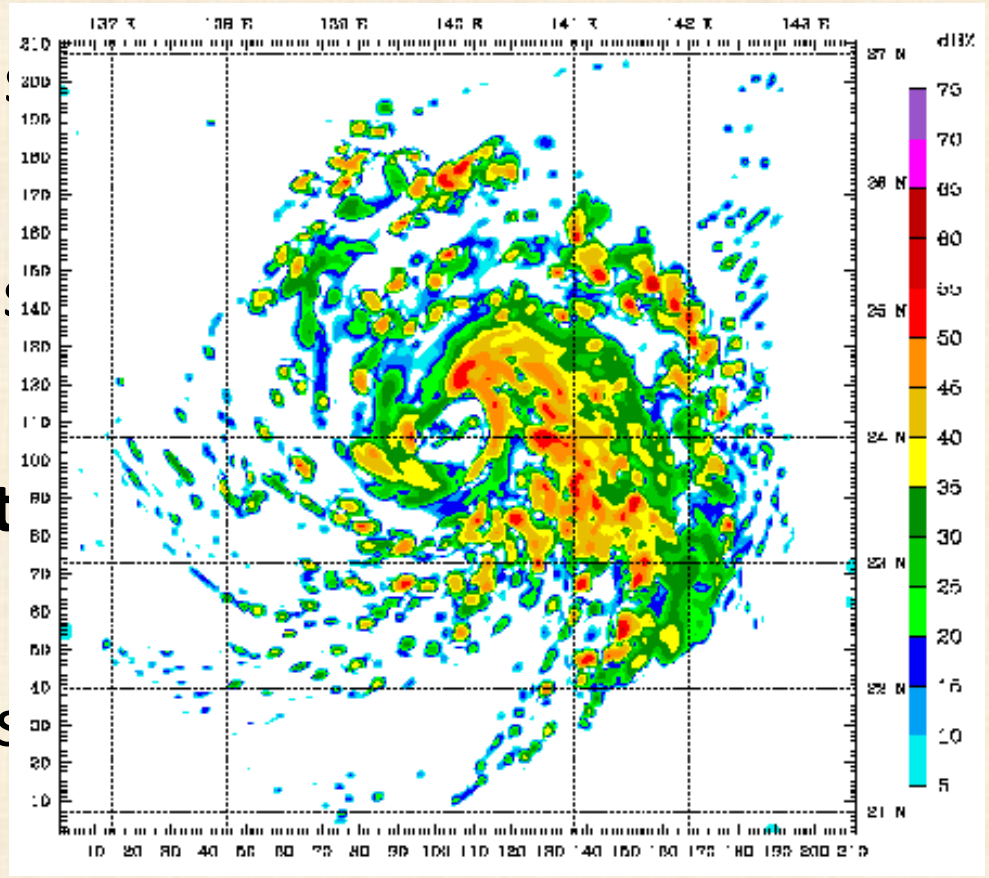
- Gam
- **Selec**
- Com
- Test
- Computer security: key generation, nonces,...

Where do we use randomness

- Games, lottery, cassino, bingo...
- Selection of judges and jurors
- **Computer simulations**
- Test of hypotheses - randomized controlled trial
- Computer security: key generation, nonces,...

Where do we use randomness

- Games, lottery, cas
- Selection of judges
- **Computer simulat**
- Test of hypotheses
- Computer security: key generation, nonces,...

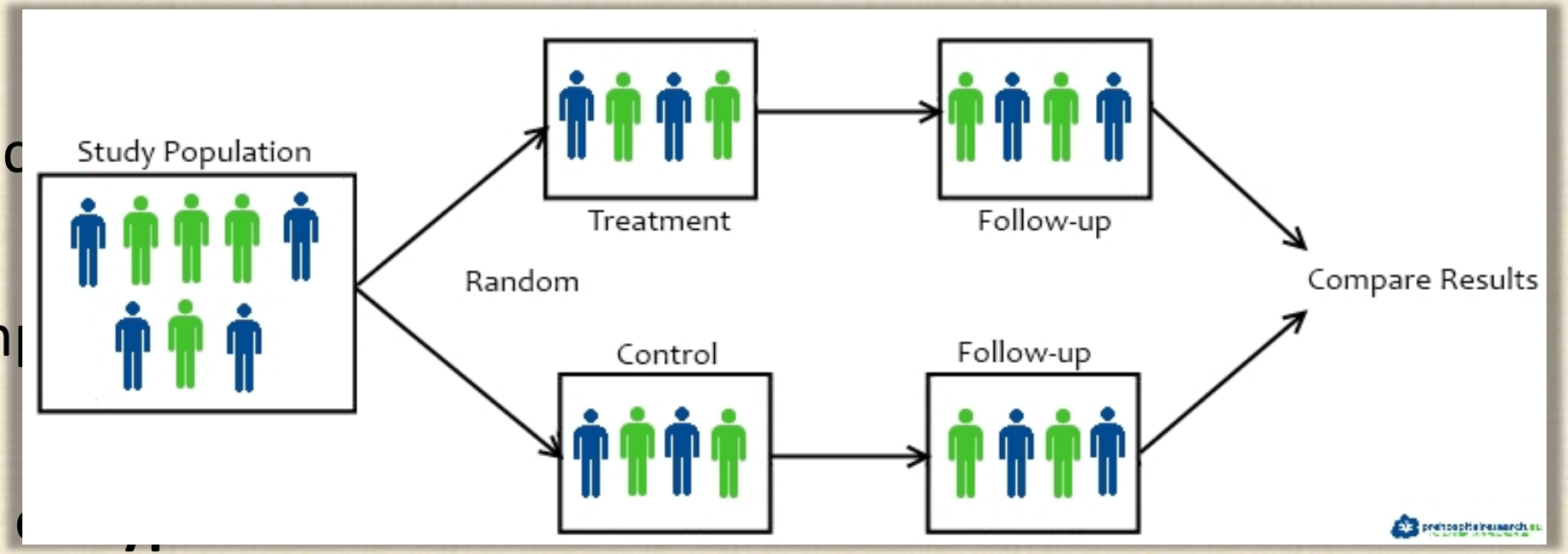


Where do we use randomness

- Games, lottery, cassino, bingo...
- Selection of judges and jurors
- Computer simulations
- **Test of hypotheses - randomized controlled trial**
- Computer security: key generation, nonces,...

Where do we use randomness

- Games, lottery, cassino, bingo...
- Selec
- Comp
- **Test**
- Computer security: key generation, nonces,...



Where do we use randomness

- Games, l
- Selection
- Compute
- **Test of hy**
- Computer security: key generation, nonces,...

Where do we use randomness

- Games,
- Selectio
- Comput
- **Test of**
- Computer security: key generation, nonces,...

Where do we use randomness

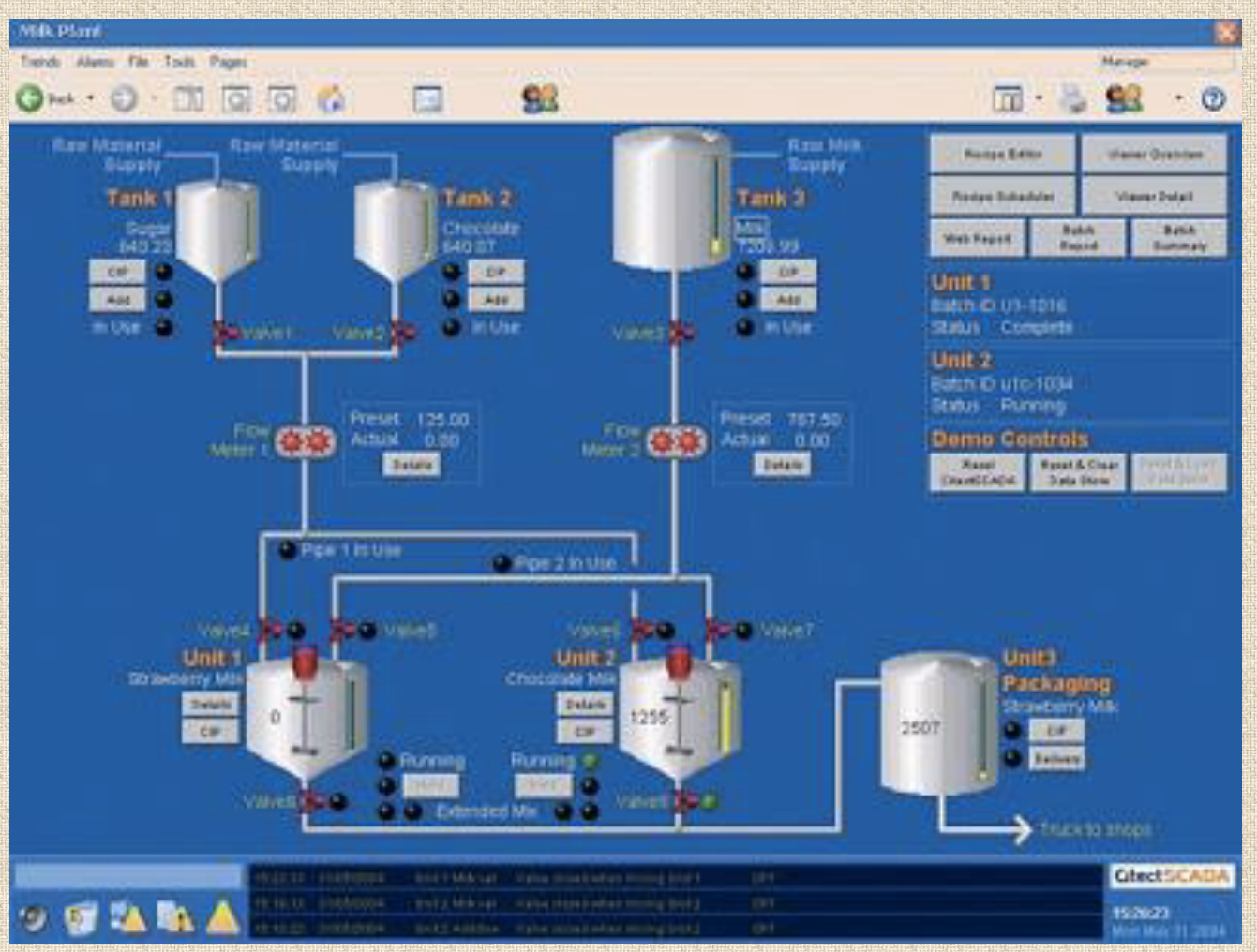
- Games, lottery, cassino, bingo...
- Selection of judges and jurors
- Computer simulations
- Test of hypotheses - randomized controlled trial
- **Computer security: key generation, nonces,...**

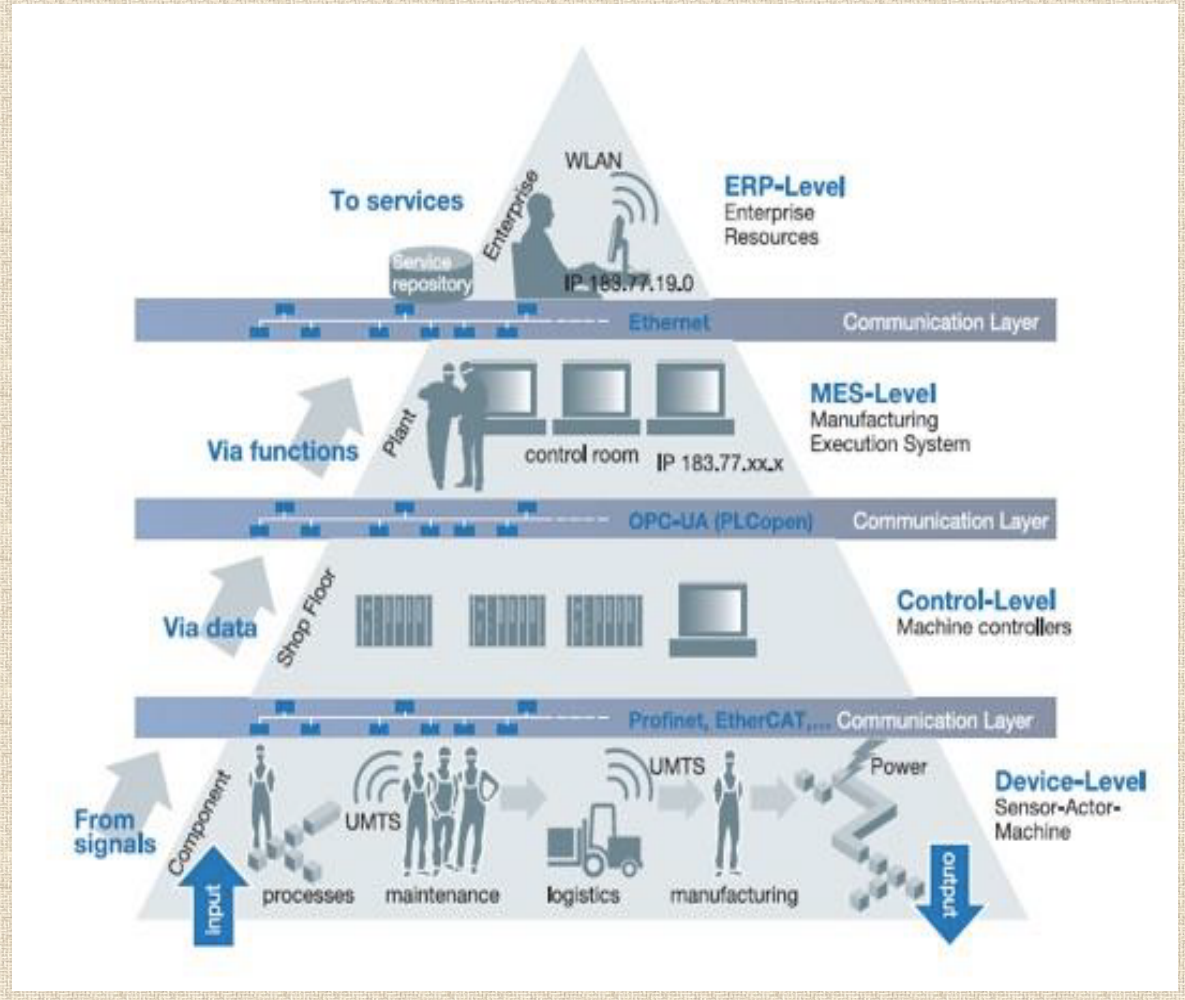
Whe

- Game
- Select
- Comp
- Test o
- **Computer security: key generation, nonces,...**

Why randomness in the industry

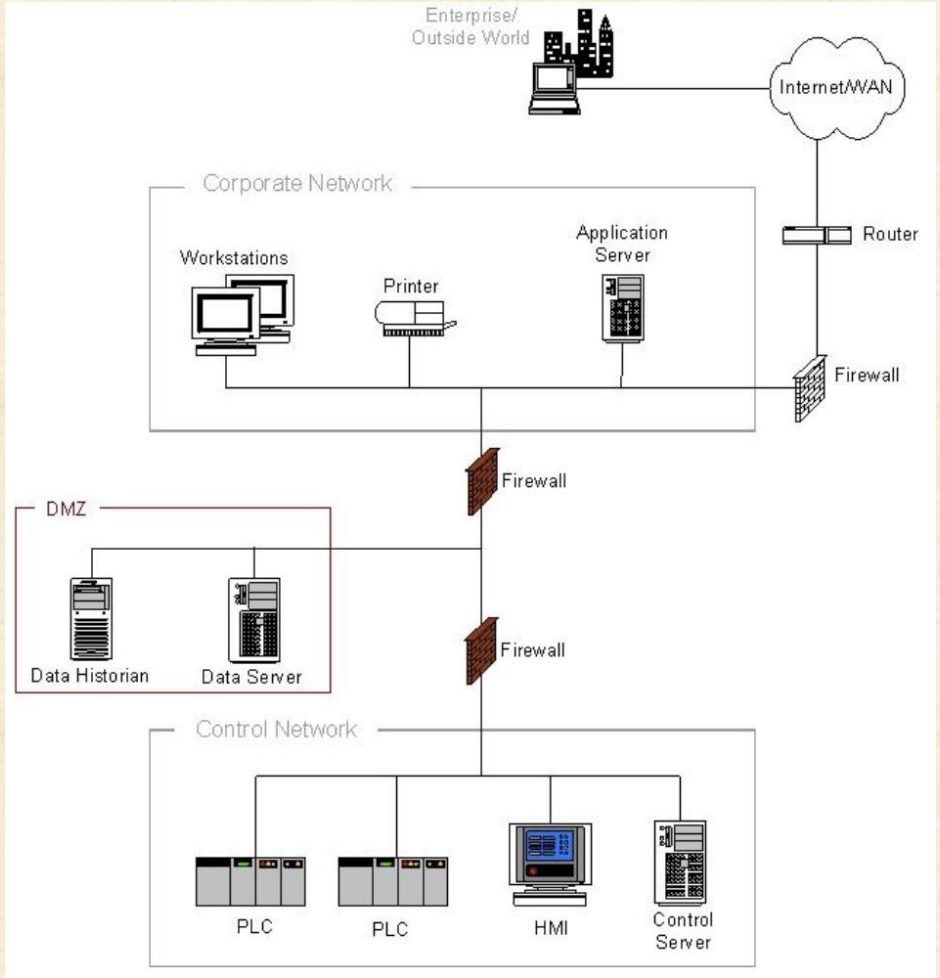
- Current industrial plants are complex computer systems...
  ... and vulnerable to the same kind of attacks

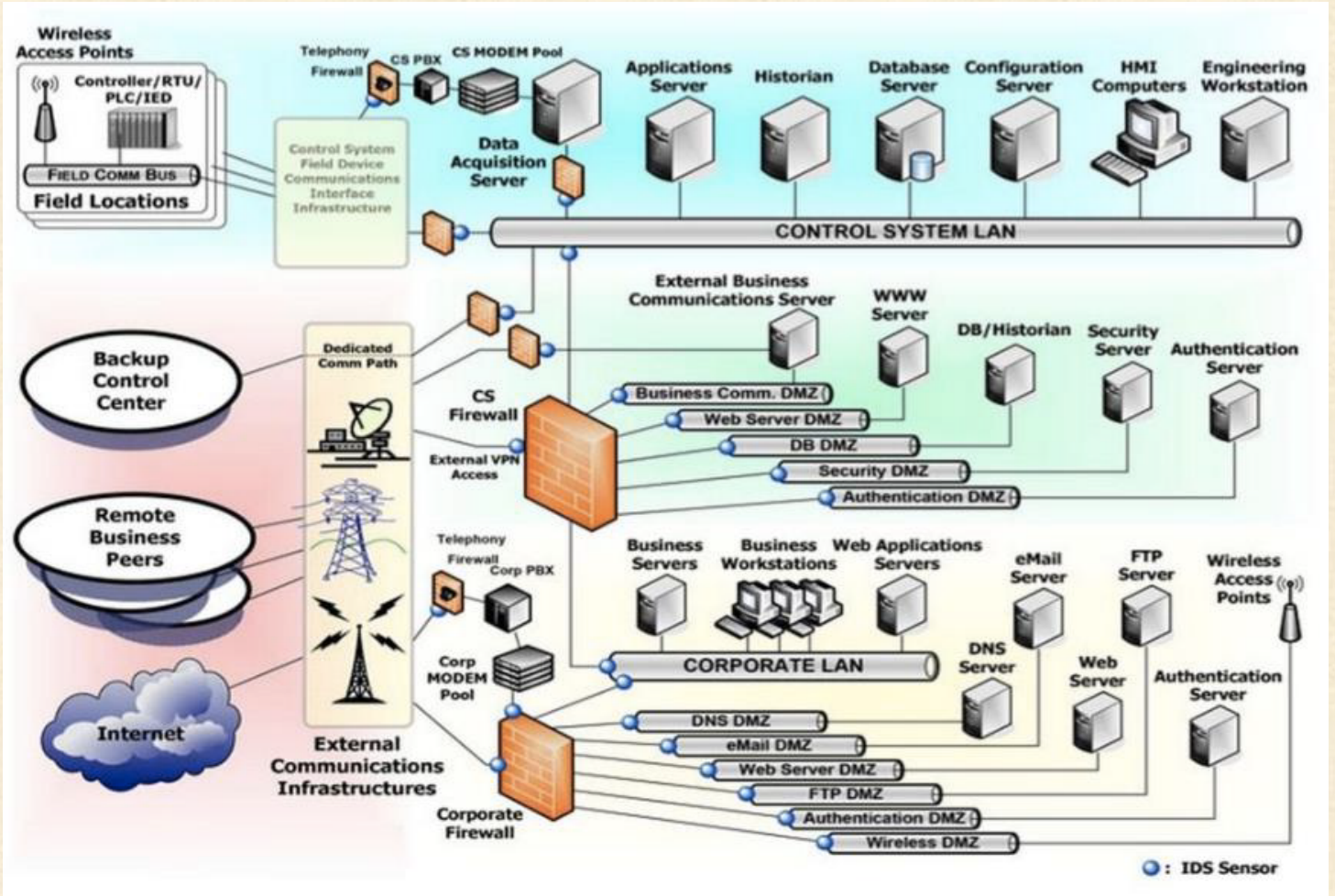




Why randomness in the industry

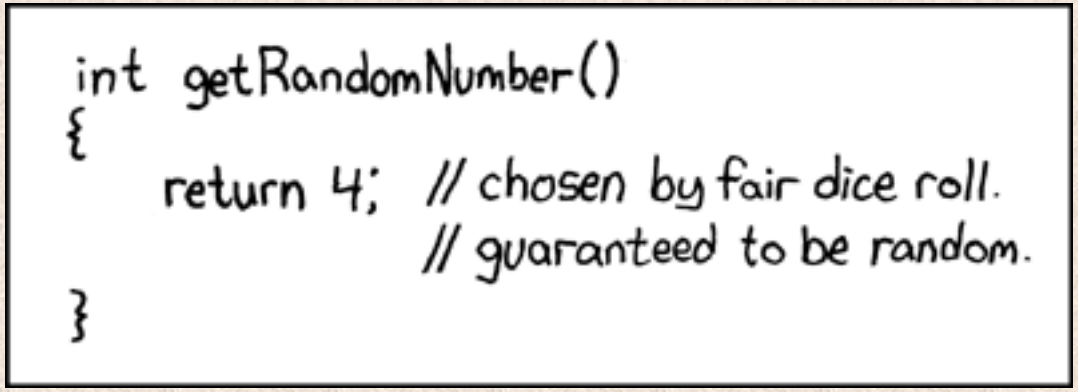
- Current industrial plants are complex computer systems...
  ... and vulnerable to the same kind of attacks





In practice: how do we get random numbers?

- Step 1: find a ”good” source of ”physical” randomness
  - Show that your source is unpredictable (in practice) and has no patterns
    - Statistic tests, implementation analysis
  - If you have a quantum source, great!
  - If you can prove quantumness, explendid! (Bell tests!)
- Step 2: post-process the obtained numbers using a PRNG
  - Simple way to get uniform distribuitions frmo non-uniform ones



Attacks on random numbers

- Critical activities dependent on good random numbers
- Attacks to random number generators
  - Netscape SSL
  - Microsoft Windows 2000/XP
  - MIFARE NXP CRYPTO-1
  - Android Bitcoins
  - Playstation 3
  - NIST SP 800-90
- Poor randomness can have critical impact!

Cryptanalysis of the Random Number Generator of the Windows Operating System

Leo Dorrendorf
School of Engineering and Computer Science
The Hebrew University of Jerusalem
91904 Jerusalem, Israel
dorrel@cs.huji.ac.il

Zvi Gutterman
School of Engineering and Computer Science
The Hebrew University of Jerusalem
91904 Jerusalem, Israel
zvikag@cs.huji.ac.il

Benny Pinkas*
Department of Computer Science
University of Haifa
31905 Haifa, Israel
benny@pinkas.net

November 4, 2007

Abstract

The pseudo-random number generator (PRNG) used by the Windows operating system is the most commonly used PRNG. The pseudo-randomness of the output of this generator is crucial for the security of almost any application running in Windows. Nevertheless, its exact algorithm was never published.

We examined the binary code of a distribution of Windows 2000, which is still the second most popular operating system after Windows XP. (This investigation was done without any help from Microsoft.) We reconstructed, for the first time, the algorithm used by the pseudo-random number generator (namely, the function `CryptGenRandom`). We analyzed the security of the algorithm and found a non-trivial attack: given the internal state of the generator, the previous state can be computed in $O(2^{23})$ work (this is an attack on the forward-security of the generator, an $O(1)$ attack on backward security is trivial). The attack on forward-security demonstrates that the design of the generator is flawed, since it is well known how to prevent such attacks.

We also analyzed the way in which the generator is run by the operating system, and found that it amplifies the effect of the attacks. The generator is run in user mode rather than in kernel mode, and therefore initial values of p by whatever value process runs a dif system generated it. The result of reveal 128 Kbytes The implicatio be used to learn a

*Research supported in

Android bug batters Bitcoin wallets

Old flaw, new problem

By Richard Chirgwin 12 Aug 2013 at 00:43 9 SHARE ▼

Users of Android Bitcoin apps have woken to the unpleasant news that an old pseudo random number generation bug has been exploited to steal balances from users' wallets.

The Bitcoin Foundation's announcement, here, merely states that an unspecified component of Android "responsible for generating secure random numbers contains critical weaknesses, that render all Android wallets generated to date vulnerable to theft."

USENIX Security Symposium. San Jose, CA. 31 July 2008.

Reverse-Engineering a Cryptographic RFID Tag

Karsten Nohl and David Evans
University of Virginia
Department of Computer Science
{nohl,evans}@cs.virginia.edu

Starbug and Henryk Plötz
Chaos Computer Club
Berlin
starbug@ccc.de, henryk@ploetzli.ch

Abstract

The security of embedded devices often relies on the secrecy of proprietary cryptographic algorithms. These algorithms and their weaknesses are frequently disclosed through reverse-engineering software, but it is commonly thought to be too expensive to reconstruct designs from a hardware implementation alone. This paper challenges that belief by presenting an approach to reverse-engineering a cipher from a silicon implementation. Using this mostly automated approach, we reveal a cipher from an RFID tag that is not known to have a software or micro-code implementation. We reconstruct the cipher from the widely used Mifare Classic RFID tag by using a combination of image analysis of circuits and protocol analysis. Our analysis reveals that the security of the tag is even below the level that its 48-bit key length suggests due to a number of design flaws. Weak random numbers and a weakness in the authentication protocol allow for pre-computed rainbow tables to be used to find any key in a matter of seconds. Our approach of deducing functionality from circuit images is mostly automated, hence it is also feasible for large chips. The assumption that algorithms can be kept secret should therefore to be avoided for any type of silicon chip.

Il faut qu'il n'exige pas le secret, et qu'il puisse sans inconvénient tomber entre les mains de l'ennemi.
([A cipher] must not depend on secrecy, and it must not matter if it falls into enemy hands.)
August Kerckhoffs, *La Cryptographie Militaire*, January 1883 [13]

1 Introduction

It has long been recognized that security-through-obscurity does not work. However, vendors continue to believe that if an encryption algorithm is released only as a hardware implementation, then reverse-engineering the cipher from hardware alone is beyond the capabilities of likely adversaries with limited funding and time. The design of the cipher analyzed in this paper, for example, had not been disclosed for 14 years despite more than a

ture and were able to fill in the missing details through cryptanalysis of the cipher output for known keys and inputs. This black-box approach requires some prior understanding of the structure of a cipher and is only applicable to ciphers with statistical weaknesses. The output of a sound cipher should not be statistically biased and therefore should not leak information about its structure.

Other ciphers have been disclosed through disassembly of their software implementation. Such implementations can either be found in computer software or as microcode on an embedded micro-controller. Ciphers

NIST Removes Cryptography Algorithm from Random Number Generator Recommendations

April 21, 2014

Following a public comment period and review, the National Institute of Standards and Technology (NIST) has removed a cryptographic algorithm from its draft guidance on random number generators. Before implementing the change, NIST is requesting final public comments on the revised document, *Recommendation for Random Number Generation Using Deterministic Random Bit Generators* (NIST Special Publication 800-90A, Rev. 1).

The revised document retains three of the four previously available options for generating pseudorandom bits needed to create secure cryptographic keys for encrypting data. It omits an algorithm known as Dual_EC_DRBG, or Dual Elliptic Curve Deterministic Random Bit Generator. NIST recommends that current users of Dual_EC_DRBG transition to one of the three remaining approved algorithms as quickly as possible.

In September 2013, news reports prompted public concern about the trustworthiness of Dual_EC_DRBG. As a result, NIST immediately recommended against the use of the algorithm and reissued SP 800-90A for public comment.

MEDIA CONTACT

Jennifer Huergo
jennifer.huergo@nist.gov
(301) 975-6343

ORGANIZATIONS

Information Technology Laboratory
Computer Security Division

Schneier on Security

Blog | Newsletter | Books | Essays | News | Talks | Academic | About Me

Blog >

Sony PS3 Security Broken

Sony used an ECDSA signature scheme to protect the PS3. Trouble is, they didn't pay sufficient attention to their random number generator.

EDITED TO ADD (1/13): More info.

Tags: authentication, cryptography, games, gaming consoles, random numbers, Sony, videos
Posted on January 6, 2011 at 5:52 AM • 85 Comments

Like Tweet +1

Attacks on random numbers

- Critical activities dependent of good random numbers

- Attacks to random number generators
  - Netscape SSL
  - Microsoft Windows 2000/XP
  - MIFARE NXP CRYPTO-1
  - Android Bitcoiins
  - Playstation 3
  - NIST SP 800-90

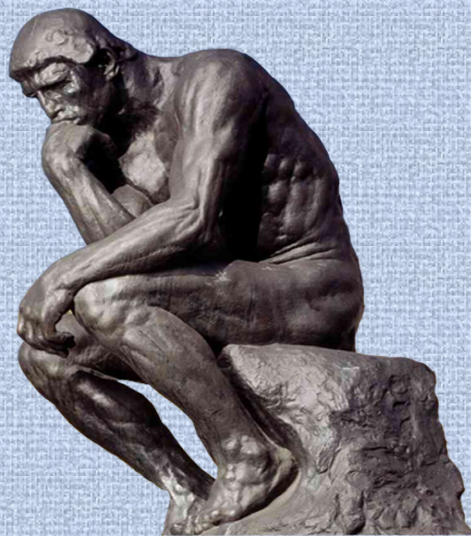
- **Poor randomness can have critical impact!**

Does randomness exist?

- Deep scientific/physical/mathematical/phylo…
  - We will not discuss it today =)

- Most of our ”sources o… …s” are…. Deterministic!
  - Dices, cards a…
  - Atmo… …mal noise sources
  - …ters random number generators…
  - …we we them anyway!

What kind of randomness do we need???



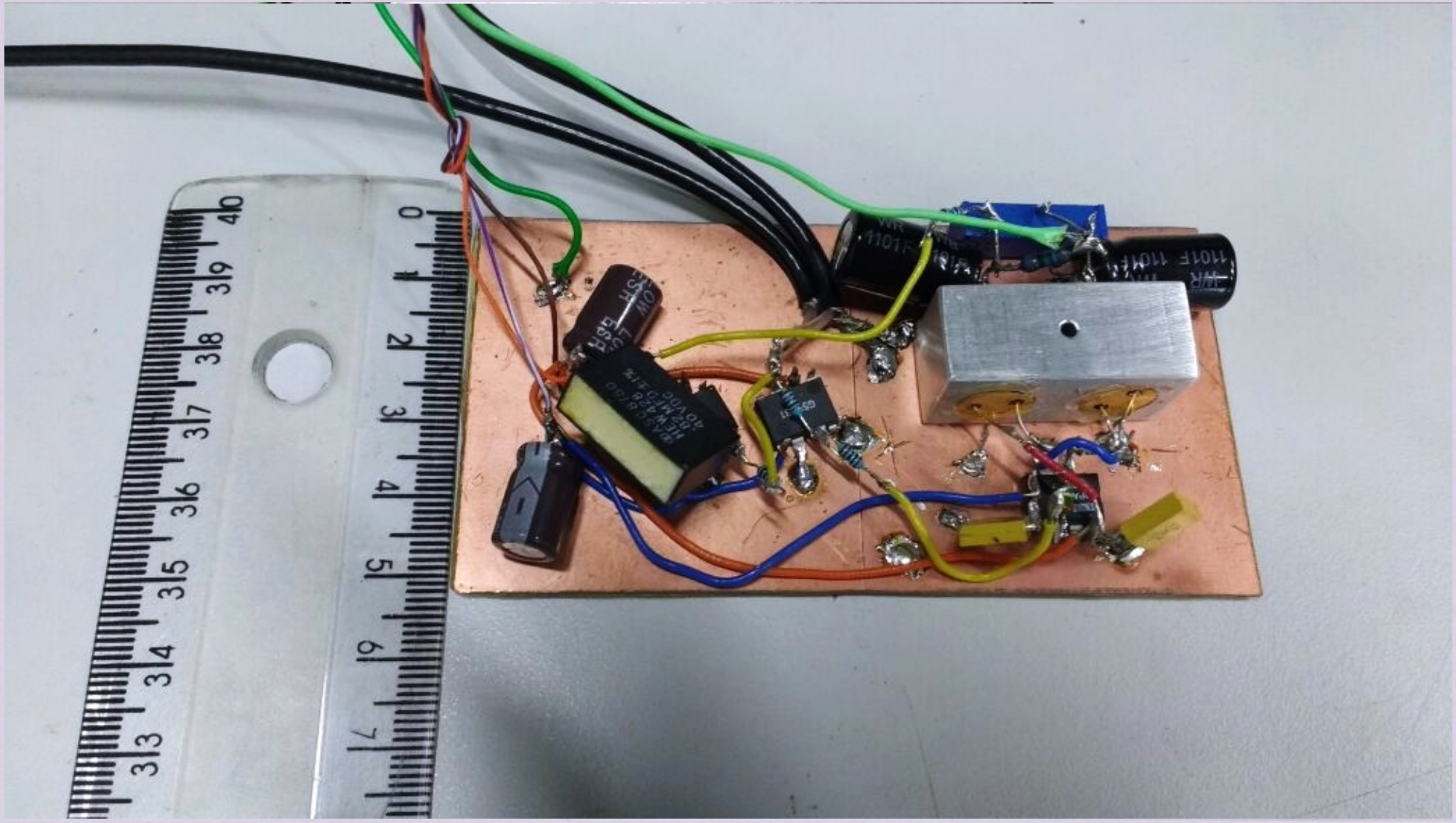
Some definitions for the concept of ”random”

- A mathematical definition
  - The output of a probabilistic event
- An information-theoretic definition
  - Information content of a string is the length of the small program that print it
  - Beautiful, but not practical theory
- A complexity-theoretic defnition
  - Pseudorandom generator: deterministic algorithm that outputs a sequence that is *indistinguishable* from a random sequence using a probabilistic polytime algorithm
  - ”randomness is in the eye of the beholder”

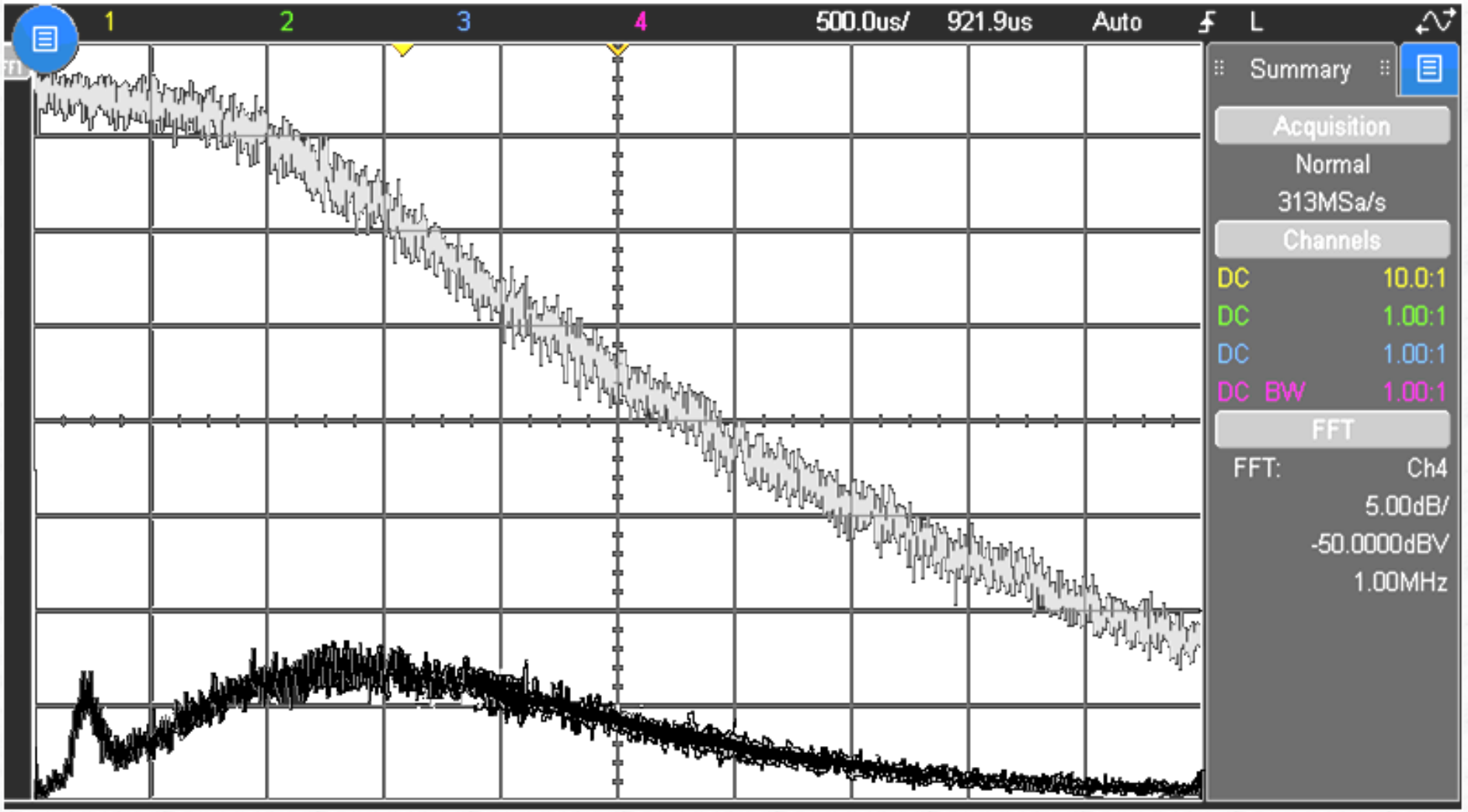
One first randomness source

Good entropy in a small package

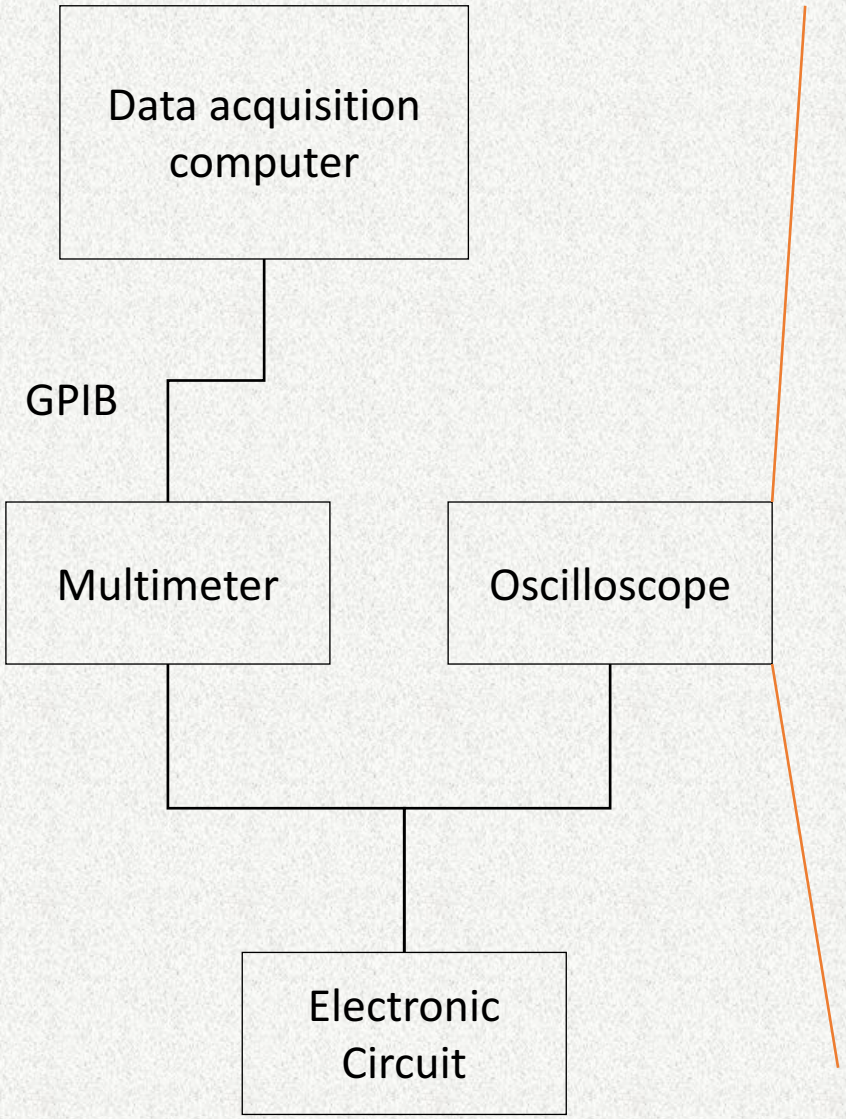
Our first prototype

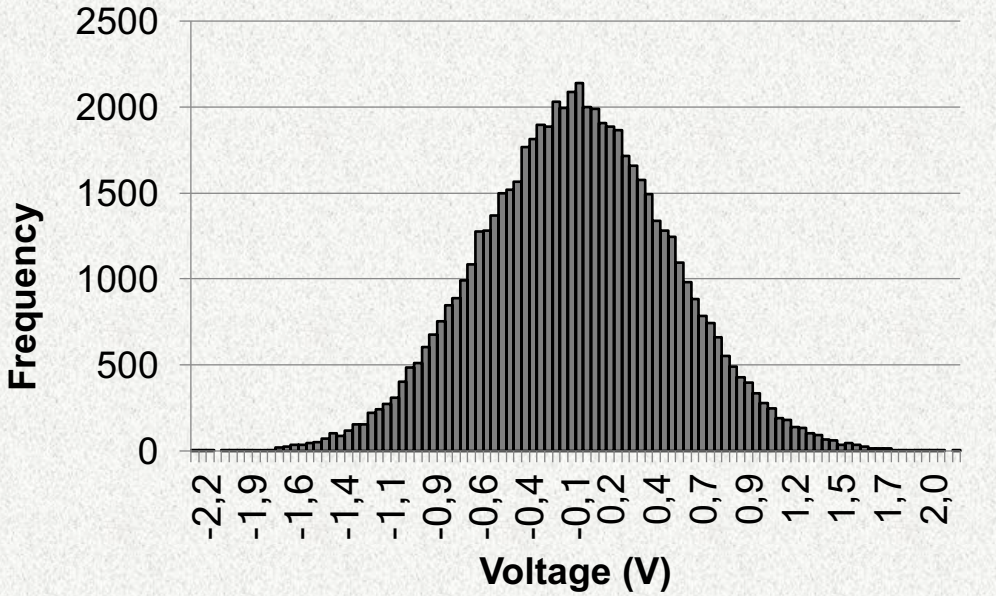


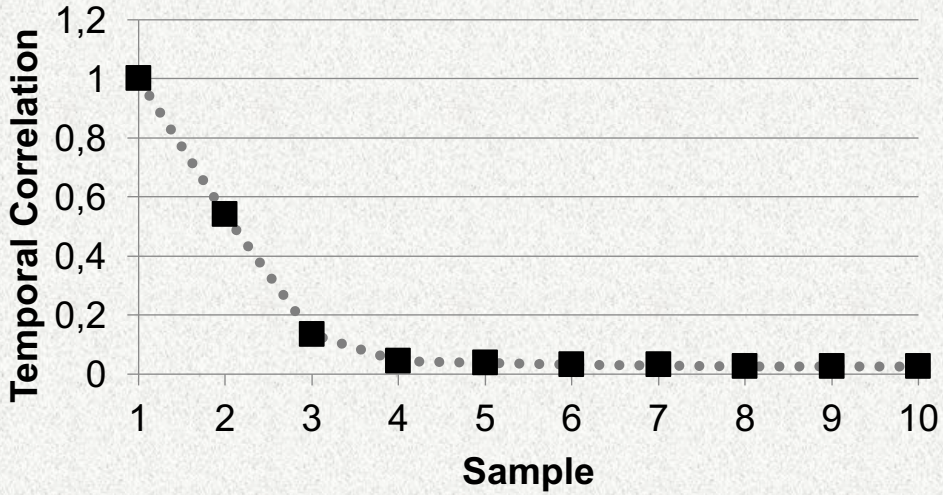
Analysis of results…



Analysis of results…





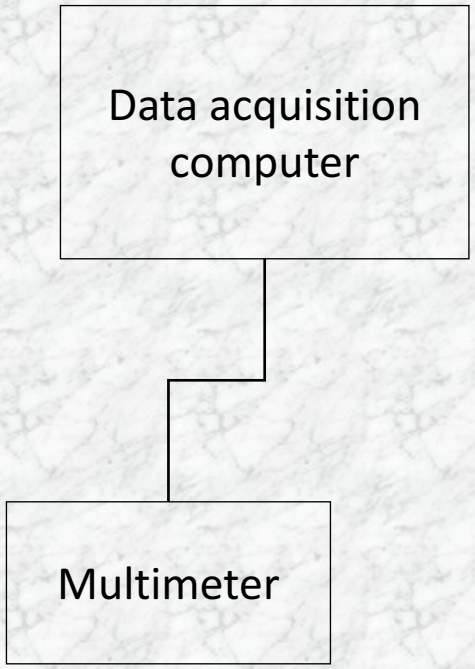


Oscilloscope sampling time: 1.6µs

Analysis of results...

- Read voltage data each 3 ms (multimeter minimum time).
- Read 8 sets of 64 voltage data (buffer limitation). Totalizing 512.
- Calculate average voltage of the full set
- Convert to bits: If voltage upper than average -> 1
  If voltage lower then average -> 0
- Concatenate bits into a 512-bits string.

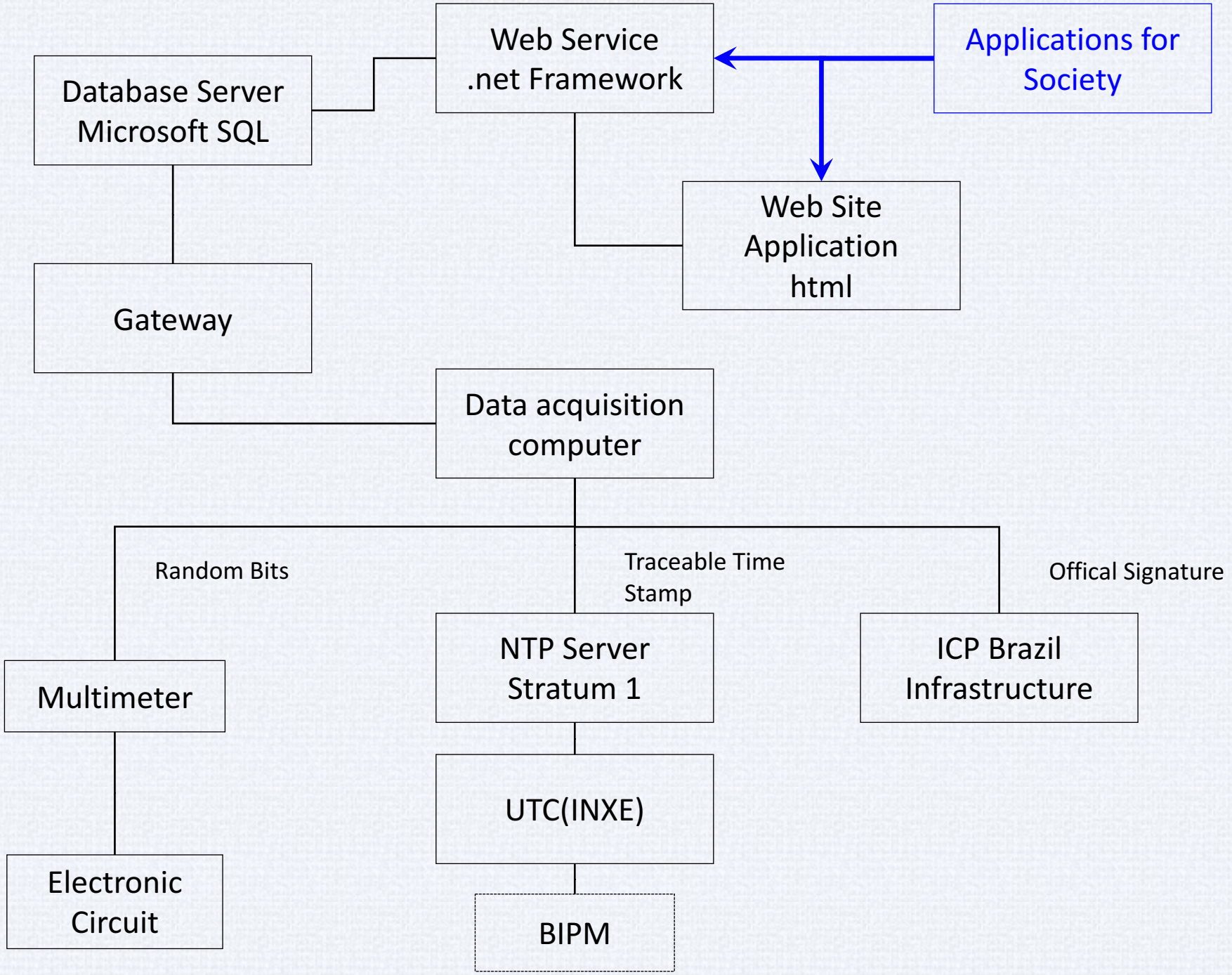
* We are exploring the symmetry of previous distribution



Randomness Test Suit

Statistical test	*p-value*	*proportion*	*result*
Frequency	0.35048	47/50	pass
Block frequency	0.000123	47/50	pass
Cumulative sum	0.171867	47/50	pass
Longest runs	0.015598	47/50	pass
Rank	0.002374	50/50	pass
FFT	0.085587	47/50	pass
Non-overlapping template	0.085587	50/50	pass
Overlapping template	0.6163	49/50	pass
Random excursions variant	0.213309	48/50	pass
Serial	0.213309	50/50	pass
Linear Complexity	0.213309	49/50	pass

Our proposed architecture


Web Service
.net Framework
Applications for
Society
Database Server
Microsoft SQL
Web Site
Application
html
Gateway
Data acquisition
computer
Random Bits
Traceable Time
Stamp
Offical Signature
Multimeter
NTP Server
Stratum 1
ICP Brazil
Infrastructure
UTC(INXE)
Electronic
Circuit
BIPM

Next Steps

Next steps

- Characterize our randonmess source according to NIST SP 800-90
- Generate multiple sources of randomness
- Devise a quantum source of randomness
- Implement a ”verifiable randomness” protocol
- Develop beacons-based security protocols
- Insert legally valid timestamp and digital signature
- Provide randomness as a service and promote applications

Thank you for your attention!

Contact: rcmachado@inmetro.gov.br

machado.work@gmail.com