OSTBAYERISCHE
TECHNISCHE HOCHSCHULE
REGENSBURG

INFORMATIK UND
MATHEMATIK

Secure Programming

Randomness

Prof. Dr. Christoph Skornia
christoph.skornia@hs-regensburg.de

Fundamentals of encryption

What do we need first for encryption?

- Random Numbers!

key gen; encryption; random tests; fuzzy tests

What is randomness? What is unpredictability?

- Unpredictability: Not possible to predict behavior!
- Randomness: No possible to predict behavior, due to principle objective reasons!

Is randomness existing at all?

- We may regard the present state of the universe as the effect of its past and the cause of its future. An intellect which at a certain moment would know all forces that set nature in motion, and all positions of all items of which nature is composed, if this intellect were also vast enough to submit these data to analysis, it would embrace in a single formula the movements of the greatest bodies of the universe and those of the tiniest atom; for such an intellect nothing would be uncertain and the future just like the past would be present before its eyes.

  —Pierre Simon Laplace, *A Philosophical Essay on Probabilities*

Is randomness existing at all?

Pseudorandomness | Lack of Knowledge | Chaos | "real" Quantum-Randomness

Randomness in Computer Science

Prediction difficult

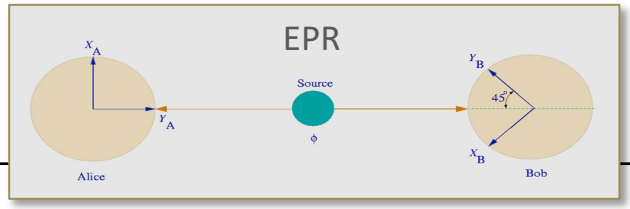
Vorhersage gundsätzlich unmöglich

Fundamentals of encryption

- Albert Einstein: “It seems hard to sneak a look at God's cards. But that He plays dice and uses "telepathic" methods... is something that I cannot believe for a single moment.”
  - this statement is based on the idea of local realism:
    - *A and B are far apart in space => external influence on A has no direct influence on B;*
    - all objects must objectively have a pre-existing value for any possible measurement before the measurement is made

- Niels Bohr: "Who is Einstein to tell the Lord what to do?”
  - John S. Bell developed a set of inequalities, which allow to design experiments to distinguish between local realism and entanglement.

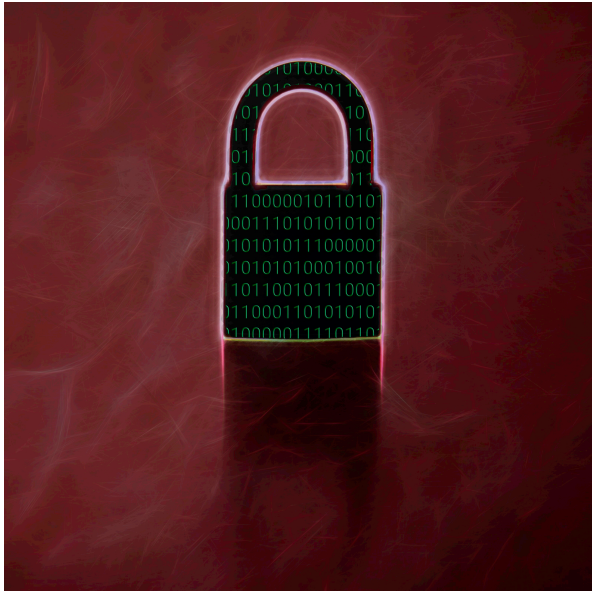
- Anthony Leggett: “… to maintain a local hidden variable theory in the face of the existing experiments would appear to require belief in a very peculiar conspiracy of nature.”



Random Numbers in IT-Security

creation of:

- keys
- nonces
- Initialisation vectors
- Salts
- canaries, ASLR
- etc.



no random numbers, no security

Randomness in Computers?

Goal:

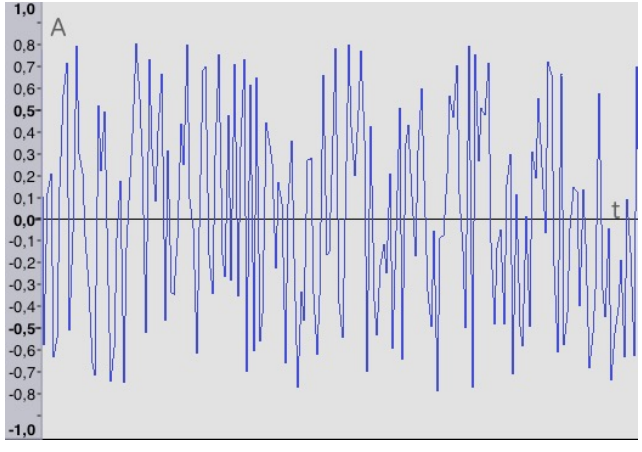
Creation of a very long series of 0 and 1, where the knowledge up to a given number, does not allow to predict future numbers.

Beispiele:

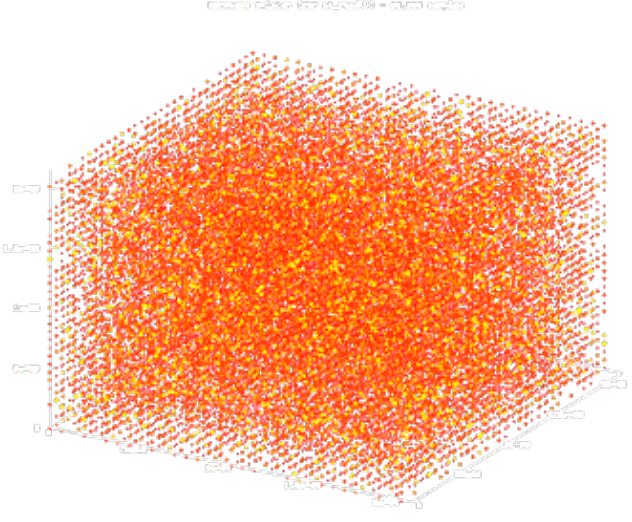
```
0101010101010101010101010101010101010101010101010101
0110000101100010011000110110001101100010011000 01
0100100101110100010100001000010110010111111001111
```

Randomness in Computers?

1. Get data from a unpredictable HW-source (e.g. diode noise)



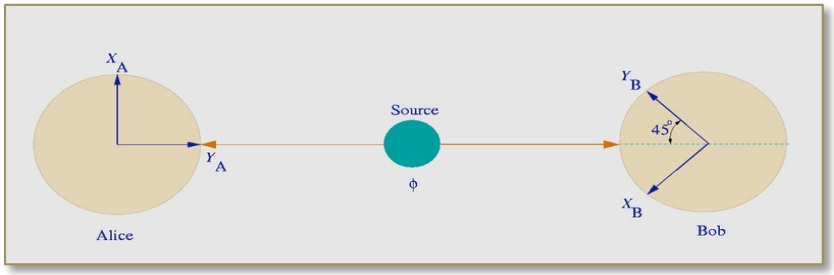
2. Use this seed of number to create a much larger number or subsequent "pseudo-random-numbers"

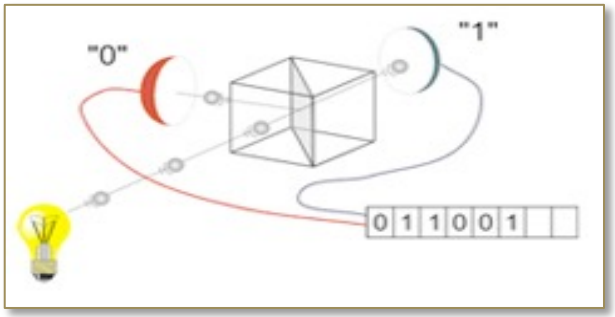


Randomness in Computers

Physical entities which behave (really) random:

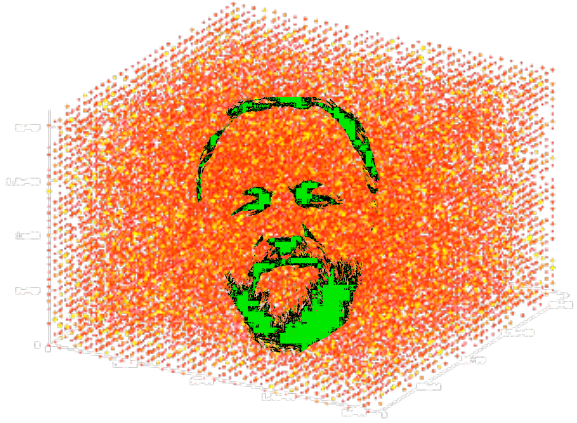
- Quantum vacuum fluctuation
- Radioactive decay
- photonic emission
- outcome of quantum measurement of equal probability
- etc.





Security Issues

- Weak generation...
  - Examples:
    - WPS on cheap WIFI-APs (Pixie-Dust)
    - Weak genearion in programming languages (e.g. rand() in C und C++)



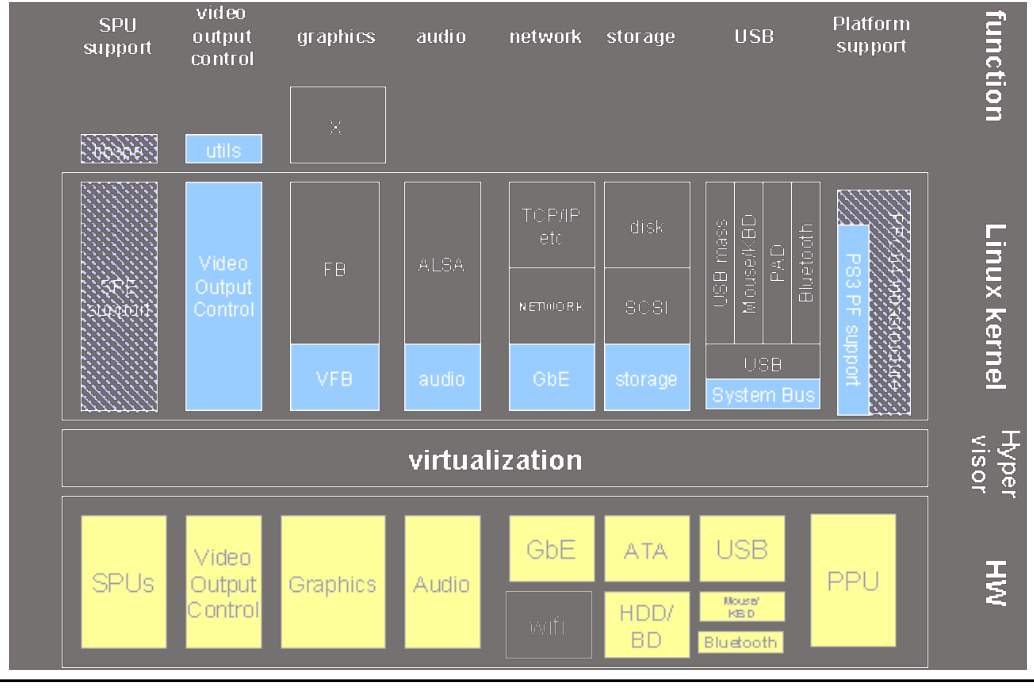
Security Issues

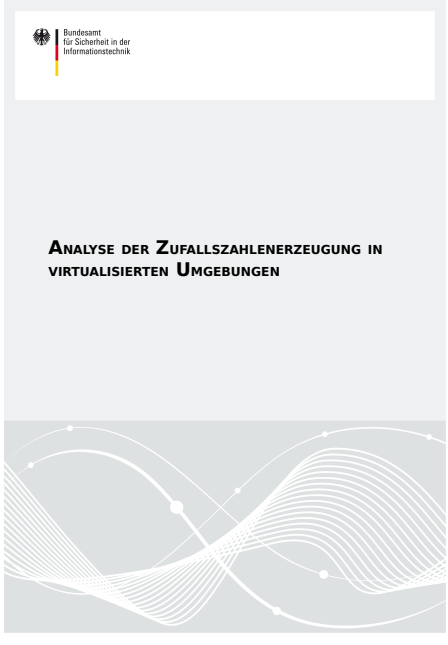
- Manipulation
  - insecure standards
    - Insecure Dual EC DRBG pushed by NSA as RNG-standard
    - 10Mio$ paid to RSA for default-config
  - Manipulation of RNG in OS
    - Malware manipulating RNG
    - Manipulation in hardware

Security Issues

- Special case: virtual machines and snapshots
  - Source of entropy not under control

tip: Wait at least 1 sec before generating random number after start-up





Fundamentals of encryption

What do we need first for encryption?

- a key which is of good (!!!) randomness
- How to get it?
  - Noncryptographic pseudo RNGs
    More properly, these are noncryptographic pseudo-random number generators.
    You should generally assume that an attacker could predict the output of such a generator.
  - Cryptographic pseudo-random number generators (CPRNGs)
    These take a single secure seed and produce as many unguessable random numbers from that seed as necessary. Such a solution should be secure for most uses as long as a few reasonable conditions are met (the most important being that they are securely seeded).
  - Entropy harvesters
    These are sometimes "true" random number generators—although they really just try to gather entropy from other sources and present it directly. They are expected to be secure under most circumstances, but are generally incredibly slow to produce data.
  - Quantum RNGs
    These are using quantum mechanical random processes to generate random numbers.
- Getting a good RNG is an art by itself, lot of weaknesses in encryption come from the non-randomness of RNG

Fundamentals of encryption

- Guidelines:
  - Postprocess entropy data with cryptographic method to remove statistical bias
  - Make sure to harvest enough data
  - If you use PRNG, make sure to proof, that they are cryptographically strong (refereed proof is best)

  - Respect:
    - Thread Safety
    - Fork Safety
    - Virtual Machines

Fundamentals of encryption

Best way of getting RN:

- use random infrastructure wherever possible
  - Unix-type infrastructures: read from /dev/random
  - Windows: CryptGenRandom()
- Use other CPRNG systems like:
  - cprng-aes
  - openssl
  - **not RC4**
  - etc…
- use external sources of random-numbers if applicable

Fundamentals of encryption

Example (Unix):

```
int rlength =  atoi(argv[1]);
unsigned char * r = (unsigned char *) malloc(rlength);


printf("\n\n/dev/urandom RNG\n");
FILE *random = fopen("/dev/urandom", "r");
fread(r, 1, rlength, random);
fclose(random);

BIO_dump_fp(stdout,r,rlength);


printf("\n\n/dev/random RNG\n");
random = fopen("/dev/random", "r");
fread(r, 1, rlength, random);
fclose(random);

BIO_dump_fp(stdout,r,rlength);
```

Fundamentals of encryption

Example (Windows):

```
#include <windows.h>
#include <wincrypt.h>

static HCRYPTPROV hProvider;
HCRYPTPROV   hCryptProv;
BYTE         pbData[16];


CryptGenRandom(hProvider, 8, pbBuffer)
```

Example (openssl):

```
#include <openssl/rand.h>

int rlength =  atoi(argv[1]);
unsigned char * r = (unsigned char *) malloc(rlength);

RAND_bytes(r, rlength);
BIO_dump_fp(stdout,r,rlength);
```

thanks for your interest

to be continued