

# Heads and Tails @ b-it ss 2009

A students  
lecture notes  
set with  
OpenOffice.org Writer  
Graphics: Ink Scape

Text and canvas adopted from Prof. Dr. Joachim von zur Gathen

## Cutline:

**bg gray 30%**

Date of lecture: dd.mm.yy

**Bold(:)**

Headline

Underlined:

following theorem, lemma or definition

... Underlined ...

important

*Italic bold*

defined item

$\hat{A}$

A is an algorithm

**Question:**

What is a „random-bit“? That is a nonsense-question. The right question is: What is a random-bit generator? (machine, random variable, distribution)

What is the use of randomness? -Cryptography

**Example:** Quicksort                      8      3      5      2      1      7

- Cost for sorting n elements
- number of comparisons is  $O(n^2)$  in worst-case and  $O(n \log n)$  in average-case when using deterministic choice of pivots

random choice of pivot:  $O(n \log n)$  worst-case

advantages of probabilistic algorithms:

- simplicity
- speed

**Example:** identity task:  $a, b, c \in \mathbb{R}^{n \times n}$ : is  $a \cdot b = c$  ?

Calculation has cost  $O(n^3)$ . But there is a random algorithm with cost  $O(n^2)$ :

Use a random-vector  $\vec{x}$  and check: is  $a \cdot b \cdot \vec{x} = a \cdot (b \cdot \vec{x}) = c \cdot \vec{x}$  ? As you see the calculation cost is now  $O(n^2)$ :

$$\underbrace{a \cdot \underbrace{(b \cdot \vec{x})}_{O(n^2)}}_{O(n^2)} = \underbrace{c \cdot \vec{x}}_{O(n^2)}$$

**answer correct**  $\leftarrow c = ab \neq c \rightarrow$  **answer correct** with certain probability.

**Probabilistic (randomized) algorithms**

caution: this is not average-case analysis.

Fingerprinting technique Freivalds ~1970s.

Identify a large object by a small amount of information. („hash function“)

Not cryptographic, just information theoretic.

Generic example: Given a polynomial  $f \in R[x_1, \dots, x_n]$  implicitly.  $R[x_1, \dots, x_n]$  is the **ring** of polynomials in  $x_1, \dots, x_n$ . That means  $\forall i, j, k \in (1, \dots, n)$  :

- $x_i + x_j \in R[x_1, \dots, x_n]$  (closure for addition)
- $\exists 0: x_i + 0 = x_i$  (additive identity)
- $x_i + x_j = x_j + x_i$  (additive commutativity)
- $\exists x_i': x_i + x_i' = 0$  (additive inverse)
- $x_i + (x_j + x_k) = (x_i + x_j) + x_k$  (associativity of addition)
- $x_i \cdot x_j \in R[x_1, \dots, x_n]$  (closure for multiplication)

- $x_i \cdot (x_j \cdot x_k) = (x_i \cdot x_j) \cdot x_k$  (associativity for multiplication)
- $\exists 1 : x_i \cdot 1 = x_i$  (identity for multiplication)
- $x_i(x_j + x_k) = x_i x_j + x_i x_k$  (distributivity)
- an inverse operation for multiplication is NOT needed.

Now we discuss the question, how one can efficiently test:

Is  $f=0$

1. arithmetic circuit:

input: constants,  $x_1, \dots, x_n$

operations: +, -, \*

output: one polynomial  $\sum a_k x^k$

**Definition degree:** for  $f = x_1^{p_1} + \dots + x_n^{p_n}$   $degree(f) = \max_{1 \leq i \leq n} (p_i)$ .  
(wikipedia)

2. For example f could be the determinant of linear polynomials:

$$f = \det \begin{pmatrix} x_1 + x_3 & \dots & x_2 - 5x_3 + 2 \\ x_3 + x & \dots & 5x_2 + 1 \\ \vdots & \vdots & \vdots \end{pmatrix}$$

All entries are linear polynomials of  $x_1, \dots, x_n$ .

What complexity has the solution of this problem? How many terms can I have? Degree

$\leq n$  in each variable:  $(n+1)^n$  But we don't care about this. We want to use an

probabilistic algorithm: substitute random values  $a_1, \dots, a_n$  for  $x_1, \dots, x_n$  and test if

$$(a_1, \dots, a_n) \stackrel{?}{=} 0$$

$f=0 \Rightarrow f(a_1, \dots, a_n)=0$ , because  $f=0 \forall x_i$ , especially all  $a_i$

$f \neq 0 \Rightarrow \text{prob}(f(a_1, \dots, a_n) \neq 0) > ?$

simple case: 1 variable:  $f \in R[x], \text{deg } f \leq n, f \neq 0$

$$\text{prob}(f(a)=0) = \frac{\#\{a : f(a)=0\}}{\#\{all a\}}$$

example:

$\mathbb{Z}_4 = \{0, 1, 2, 3\}$ .  $f : \mathbb{Z} \rightarrow \mathbb{Z}_4$ ,  $f(n) := 2n = 0, 2, 0, 6, 0, 10, 0, 14, 0, 18, \dots$  and

$$\text{prob}(f(a)=0) = \frac{n}{2n} = 50\%$$

example2:

$R = \mathbb{Z}_{15}$ ,  $f = x^2 - 1$ , roots:  $1, -1 = 14, 4, 11 = -4$

I suppose there are no other roots, because  $x \in \mathbb{Z}_{15}$ .

Definitions:

(i) **ring:** +, -, \*, 0, 1 with „usual“ rules, for example:

$R_1 = \mathbb{Z}$  ,  $R_2 = \mathbb{Z}_{15}$  ,  $R_3 = \mathbb{Z}[x]$  ,  $R_4 = R[x_1, x_2, x_3]$  ,  $R_5 = n \times n$  integer matrices (*not commutative*)

(ii)  $a \in R$  **zerodivisor**:  $a \neq 0$  ,  $\exists b \in R$  ,  $b \neq 0$  ,  $ab = 0$  . A zerodivisor is a multiplicative element  $a$  that is non-zero, and there is another element  $b$  (non-zero) which makes a zero by multiplication.

Example:  $a = 3 \in \mathbb{Z}_{15}$  :  $3 \cdot 5 = 15 = 0$  in  $\mathbb{Z}_{15}$  .

$$R_5 : \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} = 0$$

$\neq 0$                        $\neq 0$

(iii)  $R$  is an integral domain.  $\Leftrightarrow R$  has no zerodivisors.

Theorem:

$R$  integral domain,  $f \in R[x]$ ,  $\deg f \leq n$ ,  $f \neq 0 \Rightarrow f$  has at most  $n$  roots in  $R$  .

Proof:

Induction on  $n \geq 0$  :  $n = 0$  **CORRECT**

$n \geq 1$  : take a root  $a$  :  $f = \underbrace{(x-a)}_{\deg=1} \cdot \underbrace{g(x)}_{\deg < 1} + \underbrace{r}_{\deg < 1}$  // division by remainder  $r$  .

//  $\deg(g) < n$  ;  $(x-a)$  is split  $n$ -times from  $g$ !

substitute  $a$  for  $x$  :

$$0 = f(a) = (a-a) \cdot g(a) + r = r$$

$$f = (x-a) \cdot g \quad (6)$$

claim:  $b \in R, b \neq a, \underbrace{f(b)}_{(b-a) \cdot g(b)} = 0 \Rightarrow g(b) = 0$

by induction:  $g$  has  $\leq n-1$  roots. **QED.**

Notation: finite **probability distribution**  $D$  on  $A$  :

When  $A$  is a finite set:  $D: A \rightarrow \mathbb{R}$  , one can write  $\text{prob}(a)$  for  $D(a)$  :

$$0 \leq \text{prob}(a) \leq 1 \quad , \quad \sum_{a \in A} \text{prob}(a) = 1 \quad .$$

The **uniform distribution**  $U_A$  on  $a$  is defined:  $\forall a \in A : \text{prob}(a) = \frac{1}{\#A}$

$$x \leftarrow_R D \quad , \quad (x \leftarrow_R A) = (x \leftarrow_R U_A)$$

Schwartz-Zippel Lemma, Schönhage, Lipton

„nonzero preservation lemma“

Theorem:

$R$  be an integral domain,  $f \in R[x_1, \dots, x_n]$  ,  $\deg f \leq m$  ,  $f \neq 0$  ,  $S \subseteq R$  finite,  $s = \#S$  .

(i)  $f$  has  $\leq n s^{n-1}$  roots in  $S^n$  .

(ii)  $\text{prob}(f(a) = 0 : a \leftarrow_R S^n) \leq \frac{m}{s}$  .

Proof: Induction on  $n \geq 1$  .  $n = 1$  **CORRECT.**

$n \geq 2$  :  $f(x) = g_k x_n^k + g_{k-1} x_n^{k-1} + \dots + g_0 x_n^0$  ,  $g_0, \dots, g_k \in R[x_1, \dots, x_{n-1}]$  ,  $a \in S^{n-1}$  ,  $g_k(a) \neq 0$  (7),  $g_k \neq 0$  ,  $0 \leq k \leq m$  ,  $\deg g_k \leq m-k$  (8)

$$\#(\text{roots of } g_k \in S^{n-1}) \leq (m-k) s^{n-2} \quad (9)$$

$$\#(\text{common roots of } f \text{ and } g_k \in S^n) \leq (m-k) s^{n-1} \cdot s = (m-k) s^{n-1} \quad \dots$$

$$\#(\text{roots } \in S \text{ of } g_k(a) \cdot x_n^k + \dots + g(a)) \leq k \dots$$

$$a \in S^n$$

$$f(a) = 0 \Rightarrow \underbrace{(g_k(a) \neq 0, f(a) = 0)}_{\leq k \cdot s^{n-1}} \vee \underbrace{(g_k(a) = 0)}_{\leq (m-k)s^{n-1}}$$

together  $\leq m \cdot s^{n-1}$

### Polynomial identity testing

Given  $f \in R[x_1, \dots, x_n]$  implicitly (by a „circuit“),  $\deg f \leq m$

Question: Is  $f=0$  ?

Test: choose  $S \subseteq R$  finite with  $\#S \geq 2m$

Test:  $f(a_1, \dots, a_n) \stackrel{?}{=} 0$ ,  $a_1, \dots, a_n \leftarrow_R S$

Lemma: Nonzero preservation lemma:

$f=0 \Rightarrow$  test correct.

$f \neq 0 \Rightarrow \text{prob}(\text{Test incorrect}) \leq \frac{1}{2}$

definition:

$\deg f =$  total **degree** of  $f$

$$f = \sum_{i_1, \dots, i_n} a_{i_1, \dots, i_n} x_1^{i_1} \dots x_n^{i_n}$$

$$a_{i_1, \dots, i_n} \in \mathbb{R}, \quad \deg f = \max_{i_1, \dots, i_n}$$

example:  $\deg(5x^2 + 3x_3^3 - 6x_2^4) = 4$

Finite probability space  $(A, D)$

$S \subseteq A$  : „event“

$S$  and  $T$  are **independent**, if  $D(S \cap T) = D(S) \cdot D(T)$

Example:

$A = \{1, 2, 3, 4, 5, 6\}$ ,  $D = U_A$  (uniform distribution)

$$S_1 = \text{odd} = \{1, 3, 5\}$$

$$S_2 = \text{even} = \{2, 4, 6\}$$

$$S_3 = 0 \pmod 3 = \{3, 6\}$$

$$S_4 = 0 \pmod 4 = \{4\}$$

independence	$S_1$	$S_2$	$S_3$	$S_4$
$S_1$	$D(\{1, 3, 5\})$ $\neq$ $D(\{1, 3, 5\})$ $D(\{1, 3, 5\})$	$D(\emptyset)$ $\neq$ $D(\{1, 3, 5\})$ $\cdot D(\{2, 4, 6\})$	$D(\{3\})$ $=$ $D(\{1, 3, 5\})$ $\cdot D(\{3, 6\})$	$D(\emptyset)$ $\neq$ $D(\{1, 3, 5\})$ $\cdot D(\{4\})$
$S_2$	$D(\emptyset)$ $\neq$ $D(\{2, 4, 6\})$ $D(\{1, 3, 5\})$	$D(\{2, 4, 6\})$ $\neq$ $D(\{2, 4, 6\})$ $\cdot D(\{2, 4, 6\})$	$D(\{6\})$ $=$ $D(\{2, 4, 6\})$ $\cdot D(\{3, 6\})$	$D(\{4\})$ $\neq$ $D(\{2, 4, 6\})$ $\cdot D(\{4\})$
$S_3$	$D(\{3\})$ $=$ $D(\{3, 6\})$	$D(\{6\})$ $=$ $D(\{3, 6\})$	$D(\{3, 6\})$ $\neq$ $D(\{3, 6\})$	$D(\emptyset)$ $\neq$ $D(\{3, 6\})$

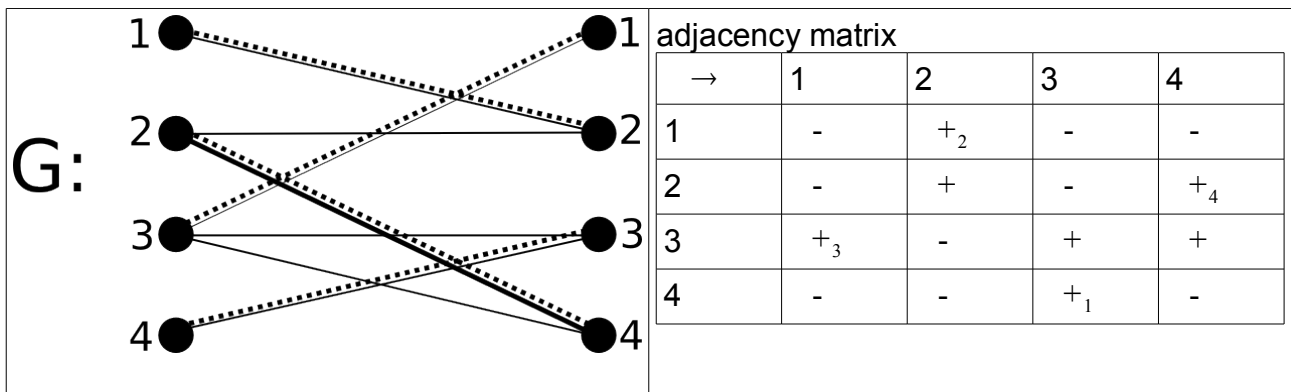
	$D(\{1,3,5\})$	$\cdot D(\{2,4,6\})$	$\cdot D(\{3,6\})$	$\cdot D(\{4\})$
$S_4$	$D(\emptyset)$ $\neq$ $D(\{4\})$ $D(\{1,3,5\})$	$D(\{4\})$ $\neq$ $D(\{4\})$ $\cdot D(\{2,4,6\})$	$D(\emptyset)$ $\neq$ $D(\{4\})$ $\cdot D(\{3,6\})$	$D(\{4\})$ $\neq$ $D(\{4\})$ $\cdot D(\{4\})$

Independence  $\leftrightarrow$  Chinese remainder theorem

$$\begin{aligned}
 S, T, U \text{ independent} &\Leftrightarrow \text{prob}(S \cap T \cap U) = \text{prob}(S) \cdot \text{prob}(T) \cdot \text{prob}(U) \\
 &\wedge \\
 &\text{prob}(T \cap U) = \text{prob}(T) \cdot \text{prob}(U) \\
 &\wedge \\
 &\text{prob}(S \cap U) = \text{prob}(S) \cdot \text{prob}(U) \\
 &\wedge \\
 &\text{prob}(S \cap T) = \text{prob}(S) \cdot \text{prob}(T)
 \end{aligned}$$

- perfect matching
- primality testing
- factoring polynomials

Bipartite perfect matching



$$\begin{pmatrix}
 0 & x_{12} & 0 & 0 \\
 0 & x_{22} & 0 & x_{24} \\
 x_{31} & 0 & x_{33} & x_{34} \\
 0 & 0 & x_{43} & 0
 \end{pmatrix} = M$$

$\det M = \sum_{\pi \in S_4} \text{sig}(\pi) M_{1\pi(1)} \dots M_{4\pi(4)}$  // Leibniz-Formula. Similar algorithm works for maximum flow

perfect matching in  $G \Leftrightarrow \pi$  with  $M_{1\pi(1)} = x_{1\pi(1)} \dots x_{4\pi(4)} \neq 0$

If  $\exists$  matching  $\Leftrightarrow \det M \neq 0$ . So the determinant has to be checked.

probabilistic algorithm for a zero-determinant-check:

$S =$  substitute random elements from  $S$  for (10)  $x_{ij}$

$S = \{-n, \dots, n\}$  for  $n$  vertices.

Calculate this instance of G.

## primality testing

Given  $N$ , try to divide by all (prime) numbers  $\leq \sqrt{N}$ . Cost:  $2^{0.5N}$

input =  $N$  in binary

$$n = \text{input length} = \log_2 N$$

$$\sqrt{N} = 2^{0.5n}$$

### Notation:

$R$  ring,  $a \in R$  **unit**  $\Leftrightarrow \exists b \in R : ab = 1$ .  $b$  is an inverse factor,  $a$  are invertible numbers.

$R^X = \{\text{units of } R\}$ ; example:  $\mathbb{Z}^X = \{\pm 1\}$ ,  $\mathbb{Q}^X = \mathbb{Q} \setminus \{0\}$

$n \in \mathbb{N}$ ,  $R = \mathbb{Z}_n$ ,  $\mathbb{Z}_n^X = \{a \in \mathbb{Z} : \gcd(a, n) = 1\}$  // gcd = greatest common divisor

$p$  prime:  $\mathbb{Z}_p^X = \mathbb{Z}_p \setminus \{0\}$

### Fermat's Little Theorem:

Conditions:  $p$  prime,  $\gcd(a, p) = 1$ . The above declarations still meet.

then  $\Rightarrow a^{p-1} = 1$

Example:

$$p = 7$$

$$1^6 = 1 \quad \Leftarrow a^{p-1} = 1$$

$$2^6 = 64 = 1$$

$$3^6 = 27^2 = (-1)^2 = 1$$

$$4^6 = (2^6)^2 = 1^2 = 1$$

$$5^6 = 25^3 = 4^3 = 2^6 = 1$$

in particular:  $2^{p-1} = 1$ . If  $2^{n-1} \neq 1$  in  $\mathbb{Z}_n \Rightarrow n$  not prime.

$S_n = \{a \in \mathbb{Z}_n^X : a^{n-1} = 1\} \subseteq \mathbb{Z}_n^X$  is subgroup.

The size of any subgroup divides the group size.

Fermat's LT  $\Rightarrow n$  prime  $\Rightarrow S_n = \mathbb{Z}_n^X$

$n$  composite  $\Rightarrow \# S_n \leq 0.5 \mathbb{Z}_n^X$  or  $\underbrace{S_n = \mathbb{Z}_n^X}_{\text{Carmichael number}}$ .

lecture 04.05.09

short note on Claus Peter Schnorr: factoring integers in polynomial time...then:

### REPETITION: Primality testing:

Given integer  $N$ , is it prime?

#### Fermat's little theorem:

Let  $N$  be prime,  $\Rightarrow a^{N-1} \equiv 1$  in  $\mathbb{Z}_N^X$ .

„ $\Leftarrow$ “ is not true, see „Carmichael-number“

$$N \approx 2^{1000}$$

computing large power  $a^k$ : repeated squaring

$k=13$ , binary 1101, length  $\log_2 k$ .

$$a^2$$

$$a^2 \cdot a = a^3$$

$$(a^3)^2 = a^6$$

$$(a^6)^2 = a^{12}$$

$$a^{12} \cdot a = a^{13}$$

cost:  $\leq 2 \text{ length} \approx 2 \log_2 k$

### Fermat test:

1. Choose  $a \in_R \mathbb{Z}_N$   
[2. if  $\gcd(a, N) \neq 1$ , return  $\gcd(\dots)$  ]
3. Compute  $a^{N-1}$  in  $\mathbb{Z}_N$
4. return „composite“ if  $b \neq 1$  and „possibly prime“ otherwise.

### Theorem:

The algorithm uses  $O(n^3)$  bit operations. ( $a \cdot b$  needs  $\leq n^2$  bits multiplied.)

Where  $n = \log_2 N$ .

The result „composite“ is always correct.

The result „possibly prime“ is correct with probability  $\geq \frac{1}{2}$ , if  $N$  is prime or a Carmichael number.

Idea:  $\sqrt{1}$  :  $\pm 1 \forall N$  prime in  $\mathbb{Z}_N$ .

And there are  $\geq 4$  elements for the case that  $N$  is no prime power.

The strong pseudo primality test was developed by: Miller 1976, Robin 1976, Solovay-Strassen 1974.

It was realized in deterministic polynomial time by Aggerval-Kayal-Saxena in 2002.

Let  $f \in \frac{F_p[x]}{\mathbb{Z}_p[x]}$  monic of degree  $n$ ,  $A \subseteq F_p$  :

$$f = \prod_{a \in A} (x - a), \text{ task: find } A \text{ from } f.$$

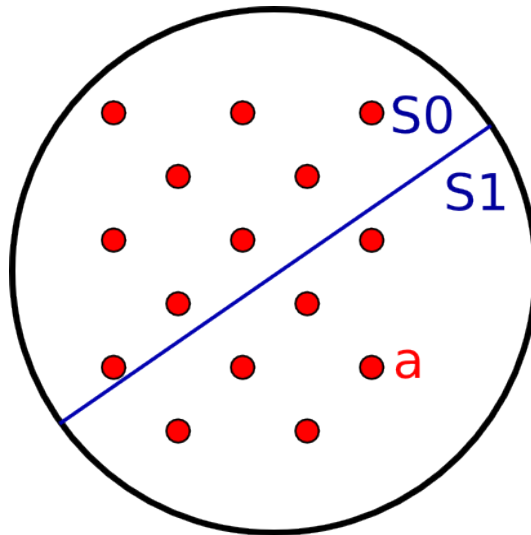
### FLT (Fermat's...)

$$a^{p-1} \equiv 1 \forall a \in \mathbb{Z}_p^X = \mathbb{Z}_p \setminus \{0\}$$

$$a^p \equiv a \forall a \in \mathbb{Z}_p$$

$$x^p - x \equiv \prod_{a \in F_p} (x - a)$$

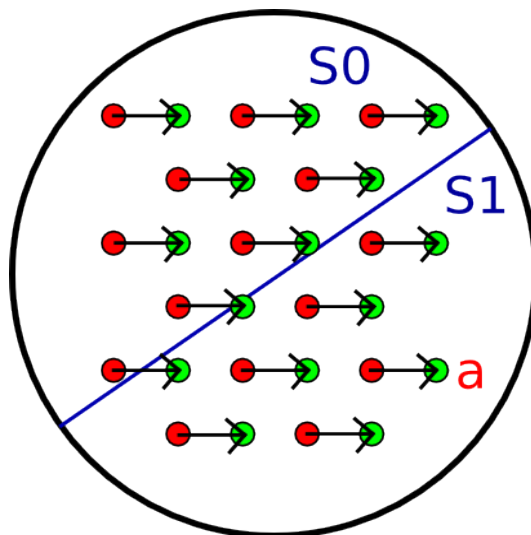
$$\equiv x(x^{p-1} - 1) \stackrel{\text{binomial}}{\equiv} \underbrace{x \left( x^{\frac{p-1}{2}} - 1 \right)}_{\prod_{a \in S_0 = \{\text{squares}\}} (x - a)} \underbrace{\left( x^{\frac{p-1}{2}} + 1 \right)}_{\prod_{a \in S_1 = \{\text{nonsquares}\}} (x - a)}$$



Assume  $0 \notin A$  .

$$\gcd(f, x^{\frac{p-1}{2}} - 1) = \prod_{a \in A \cap S_0} (x - a)$$

Shift by random number  $U \in F_p$  and split again:



Replace  $f(x)$  by  $f(x-U)$  .

Example:

$p=7$

0	1	2	3	4	5	6	$S_0$
$\notin$	$\in$	$\in$	$\notin$	$\in$	$\notin$	$\notin$	

$$f = (x-2)(x-3)(x-4)(x-5)$$

$$A = \{2, 3, 4, 5\}$$

$$A_0 = A \cap S_0 = \{2, 4\}$$

$$A_1 = A \cap S_1 = \{3, 5\}$$

shift by  $U = -1$  :

$$A_0^* = \{1, 3\}$$

$$A_1^* = \{2, 4\}$$

$$\begin{aligned}
A_{00} &= A_0^* \cap S_0 = \{1\} \\
A_{01} &= A_0^* \cap S_1 = \{3\} \\
A_{10} &= A_1^* \cap S_0 = \{2,4\} \\
A_{11} &= A_1^* \cap S_1 = \{\emptyset\}
\end{aligned}$$

**Theorem:**

Polynomials in  $F_p[x]$  can be factored probabilistically in polynomial time and in degree  $\log p$ .  
 open question: deterministic polynomial time.

**I. Probabilistic algorithms**

Higher Level View

II. Random generation: true randomness  
 Random generation: pseudo randomness

**P, BPP, NP**

Decision problems  $L \subseteq \{0,1\}^*$

Consider polynomial time algorithms (or BPP= machine)  $A$  „bounded probabilistic polynomial time“  
 with input  $x \in \{0,1\}^*$

$$A(x) = \begin{cases} \text{yes} & \text{if } x \in L \\ \text{no} & \text{if } x \notin L \end{cases} \Rightarrow L \in P \quad n = |x|$$

$A$  nondeterministic

$$A(x) = \begin{cases} \text{yes, at least once} & \text{if } x \in L \\ \text{always no} & \text{if } x \notin L \end{cases} \Rightarrow L \in NP$$

$A$  probabilistic

$$\text{prob}(A(x) = \text{yes}) = \begin{cases} \geq 1/2^* & \text{if } x \in L \\ 0 & \text{if } x \notin L \end{cases} \Rightarrow L \in BPP$$

\*He said that it could be anything else in (0,1)

Example 1: Primality testing:  $L = \{x \in \mathbb{N}; x \text{ composite}\}$

Example 2: Identity testing:  $L = \{f \in \mathbb{Z}[x_1, \dots, x_r], \text{ given as circuit : } f \neq 0\}$

$$P \subseteq BPP \subseteq NP$$

$P \neq NP$  Cook's Hypothesis  
 $L$  NP-complete:  $L \in P \Leftrightarrow P = NP$  (1)

Standard conjectures:  $P = BPP \cap co-BPP$  \*

\*complement elements of BPP

conjecture:  $NP \neq co-NP$  . ZPP\*: always correct

\*He said it was something with „zero“. Someone's mind didn't work well at this time, but I admit that there is a  $\leq \frac{1}{2}$  Chance, that it was my mind.

Since 2002 we know:  $\{primes\}$  ,  $\{composites\} \in P$  (2)

$\{factoring\ polynomial\ over\ finite\ fields\} \in ZPP$   
 $\in P??$

Impagliazzo & Kabanets: identity testing  $\in P \Rightarrow P = BPP$

## How to generate random-numbers

Hardware-based generation:

- System clock
- sector access on hard drive
- radioactive decay
- semi-conductor thermal noise
- electrical systems:
  - capacitor charge
  - oscillators
- energetic mouse movement

Drawbacks:

- expensive
- might be modified by adversary
- hard to estimate quantity and quality of randomness

lecture 11.05.09

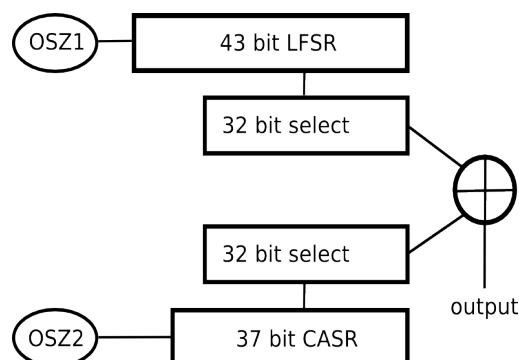
Generating randomness, entropy

Information theory: Claude Shannon 1948  
entropy, error-correcting codes

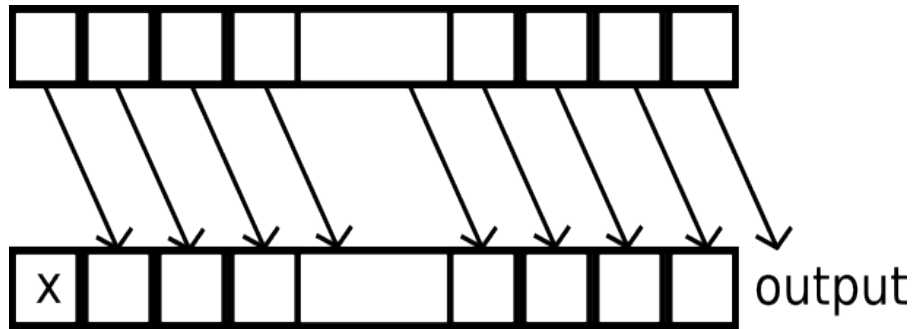
our applications:

- randomized algorithms: robust
- crypto: pseudo-randomness

Motorola Generator



linear feedback shift register LFSR:



$$x := b_1 \oplus b_2 \oplus \dots \oplus b_{43}$$

cellular automaton shift register CASR

state of cell at time  $t$  is given by states at time  $t$  (3).

Here:  $s_{t+1}(i) = s_t(i-1) \oplus s_t(i+1)$  except one place (4)

CA: Stephen Wolfram: W. Alpha Mathematica

Motorola: Sample intervals

Schindler BSI: too short

Reason: entropy too small

The sample intervals used by Motorola are too short. The BSI decided these algorithms are too unsecure to get the certificate of security.

### Entropy

Given random-bits (5) in  $\{0, 1\}$ . How to produce uniformly trits in  $\{0, 1, 2, \dots\}$  ?

I. attempt: 3 bits  $b_0, b_1, b_2$ ; output = Hamming weight mod 3 = amount of 1's mod 3.

$$\text{prob} \begin{pmatrix} 0 \\ 1 \\ 2 \\ 3 \end{pmatrix} = \begin{pmatrix} 2/8 \\ 3/8 \\ 3/8 \\ 0 \end{pmatrix}$$

II. attempt:  $b_0, b_1$

output  $b_0 + 2b_1$  if  $\leq 2$

fail if = 3

Problem is not exactly solvable, because all probabilities after  $k$  bit choices are  $\frac{m}{2^k} \neq \frac{1}{3}$  for some  $m \in \mathbb{N}$ .

Produce 3 trits via 2 bits, attempt need  $3 * 2 = 6$  bits, no (11) among them.

Need only 5 bits since  $3^3 = 27 < 2^5 = 32$ .

### Entropy

How many bits (  $B$  ) is a trit (  $T$  )?

$$\begin{aligned} T &\leq 2B \\ 2T &\leq 4B \\ 3T &\leq 5B \end{aligned}$$

$$\begin{aligned} kT \leq lB &\Leftrightarrow 3^k \leq 2^l \\ l &\geq \log_2 3^k = k \log_2 3 = k \log \left( \frac{1}{3} \right)^{-1} \end{aligned}$$

Finite set  $A$  = alphabet. Probability distribution  $p$  on  $A$  :  $\forall x \in A$  ,  $x$  occurs with probability  $p_x$  .

Encode message from  $A^*$  over  $\{0, 1\}$  . #  $A=2B$

- 1) Five bits per letter, length (n-letter message) = 5n
- 2) 8 bit Ext ASCII

Goal: minimize average length of encodings in  $A^*$

encoding

a	b	c	aa
0	1	10	11

prefix-code

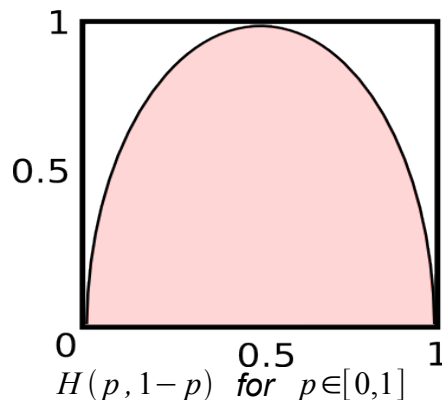
Morse code:  $e = \dots$  ,  $s = \dots$

$$\begin{aligned} A^2 &= \{aa, ab, ac, \dots, cb, cc\} \\ p &= \left\{ \frac{25}{144}, \dots, \frac{9}{144} \right\} \approx 1.573n \\ &\text{prefix for 4bit codes} \\ &\Rightarrow 000\dots110, \quad 1110, \quad 1111 \end{aligned}$$

### Definition:

Let  $p$  be a probability distribution on  $A$  . Then we define its **entropy** as

$$H(p) := \sum_{a \in A} p_a \log_2(p_a^{-1}) \quad // \text{ What is } - \sum_{a \in A} p_a \log_2(p_a)$$



$$s = \# A$$

Property:  $0 \leq H(p) \leq H\left(\frac{1}{s}, \dots, \frac{1}{s}\right) = \log_2 s$

To calculate logarithms with base 2, you can use the following law:

$$\log_b r = \frac{\log_a r}{\log_a b}, \text{ so } \log_2 s = \frac{\log_{10} s}{\log_{10} 2} \text{ or } \log_2 s = \frac{\ln s}{\ln 2}$$

Example with  $\{a, b, c\}$  :  $H(p) = 1.555$

prefix-free code  $c: A^k \rightarrow \{0, 1\}^*$

(average) length of an  $n$ -letter word  $w$ ,  $k$  dividing  $n$ ,  $w = w_1, \dots, w_n$ ,

$$= \sum_{1 \leq i \leq n} \text{length}(c(w_i)) = \lambda(w)$$

$$\text{average length} = \sum_{w \in A^n} \lambda(w) \cdot \text{prob}(w)$$

### Shannon's Theorem

For any  $\epsilon > 0$  there are  $k, c$  (type error – should be sth. else) so that average length on

$$A^n \leq n \cdot (H(p) + \epsilon)$$

(ii) For any  $k, c$  we have: average length on  $A^n \geq n \cdot H(p)$ .

Entropy of English:

$$H(\text{uniform 26 letters}) = \log_2 26 \approx 4.7$$

$$H(\text{English single letters}) \approx 4.2$$

$$\text{redundancy} = H(\text{uniform}) - H(\text{English}) \approx 0.5$$

lecture 18.05.09

### Pseudo-random generation

Idea:

- need randomness
  - randomness is expensive
  - solution: from little randomness produce deterministically a lot of pseudo-randomness.
- Pseudorandomness is algorithmically as good as randomness.

Example: linear congruential generator:

given integers  $a, b, m$

seed  $x_0 \in \mathbb{Z}_m$

for  $i = 1, 2, 3, \dots$   $x_i = a x_{i-1} + b$  in  $\mathbb{Z}_m$

This is the basis of many RAND-functions in software.

Predictor: Given  $x_0, x_1, \dots, x_{i-1}$  (and NOT  $a, b, m$ )

compute  $x_i$

Given  $x_{i-1}, x_i, x_{i+1}, x_{i+2}$  :

$$\begin{aligned} x_i &= x_{i-1} \cdot a + b & 1 & & x_i - x_{i+1} &= a(x_{i-1} - x_i) & 4 & \Rightarrow & (x_i - x_{i+1})^2 &= a(x_{i-1} - x_i)(x_i - x_{i+1}) & 6 \\ x_{i+1} &= x_i \cdot a + b & 2 & & & & & & & & & \end{aligned}$$

$$x_{i+2} = x_{i+1} \cdot a + b \quad 3 \quad x_{i+1} - x_{i+2} = a(x_i - x_{i+1}) \quad 5 \Rightarrow \quad (x_i - x_{i+1})^2 = (x_{i-1} - x_i)(x_{i+1} - x_{i+2}) \quad 7$$

Thinkpath: 1-2 = 4, 2-3=5 || 4 \* left(4) = 6 || right(5) in 6 = 7.

So: m divides  $(x_i - x_{i+1})^2 - (x_{i-1} - x_i)(x_{i+1} - x_{i+2})$

Repeat to get other multiples of m. Their gcd is also a multiple of m.

Fact:

After a polynomial number of steps we find a multiple m' of m and a', b' so that

$$x_i' = a' x_{i-1}' + b' \text{ in } \mathbb{Z}_n$$

Is the same sequence as the given m.

Modification:

Do not output  $x_i$  but remove the lower (or upper) bits of  $x_i$ .

10.....	110
---------	-----

OUTPUT

t bits

This can also be broken via computations of „short vectors in lattices“.

Quadratic, inverse congruential generators:

$$x_i = ax_{i-1}^2 + b \text{ or } x_i = \frac{a}{x_{i-1}} + b$$

They have good distribution properties.

security: unknown.

RSA generator: Given RSA parameters  $N, e, d$  and seed  $x_0$ , take  $x_i = x_{i-1}^e$  in  $\mathbb{Z}_N$ , for  $i=1,2,\dots$

Presumable secure, assuming RSA secure.

### The Littlewood generator 1953

public: integers  $n < d$

n-bit string x = seed

rational number 0.x in binary

output: d-th bit of binary representation of  $\log_2((x+i) \cdot 2^{-n}) = \log_2(x+i) - n$  for  $i=0,1,\dots$

Example:  $n=10, d=14$

i	$(x+i) \cdot 2^{-n}$		$\log_2(x+i) - 2$
0	0.011010	0111	no idea
1	0.011010	1000	he didnt
2	0.011010	1001	leave the
3	0.011010	1010	sheet on
4	0.011010	1011	projector

Distinguishers

probability distribution  $p: A \rightarrow \mathbb{R}_{\geq 0}$ , A finite.

$$\sum_{a \in A} p(a) = 1$$

Uniform distribution  $U: U(a) = \frac{1}{\#A} \quad \forall a \in A$

function  $f: A \rightarrow B$ , then we get the random variable  $X$  on  $B$  (with values in  $B$ ), which assume the value  $b \in B$  with probability  $\sum_{\substack{a \in A \\ f(a)=b}} p(a)$ . Abbreviate:  $\text{prob}(b \leftarrow_R X)$

Example: fair die

$$A = \{1, 2, 3, 4, 5, 6\} \quad . \quad f(a) = a^2 \quad ; \quad f: A \rightarrow \mathbb{R}$$

$$\text{prob}(12 \leftarrow X) = 0$$

$$\text{prob}(9 \leftarrow X) = \frac{1}{6}$$

$$E(X) = 1^2 \cdot \frac{1}{6} + \frac{2^2 \cdot 1}{6} + \frac{3^2 \cdot 1}{6} + \frac{4^2 \cdot 1}{6} + \frac{5^2 \cdot 1}{6} + \frac{6^2 \cdot 1}{6} = \frac{91}{6}$$

Given random variables  $x_1$  on  $A_1$ ,  $x_2$  on  $A_2$ , ... we get the product variable

$$X = x_1 \times x_2 \times \dots \quad \text{on} \quad A_1 \times A_2 \times \dots \quad \text{with} \quad \text{prob}(a_1, a_2, \dots \leftarrow X) = \text{prob}(a_1 \leftarrow x_1) \cdot \text{prob}(a_2 \leftarrow x_2) \cdot \dots$$

$$\pi: C \times D \rightarrow C$$

**projection**

$$\pi(X)$$

= **marginal value** of  $X$  on  $C$ .

$X$ : random variable on  $C \times D$ .

$$B = \{0, 1\}, \quad B^n = \text{set of } n\text{-bit strings.}$$

Algorithm  $A$  with output from  $B^n$  and one bit outputs.

Random variable  $X$  on  $B^n$ .

New random variable  $A(X)$ : for  $b \in B$

$$\text{prob}(b \leftarrow_R A(X)) = \sum_{x \in B^n} \text{prob}(x \leftarrow_R X) \text{prob}(b \leftarrow A(X))$$

lecture 21.05.09

## CHRISTI HIMMELFAHRT

lecture 25.05.09

### Distinguishers

#### Example:

Algorithm  $A$  takes  $X = (x_1, \dots, x_6) \in B^6$  as input and outputs  $Y = x_4$ .  $U_6$  = uniform distribution on  $B^6$ .  $E(A(U_6)) = \frac{1}{2}$ .

If  $X$  and  $Y$  are random-variables on  $B^n$ , and  $A$  an algorithm with inputs from  $B^n$  and output in  $B$ , then

$$\Delta_A(X, Y) = |E(A(X)) - E(A(Y))|$$

is the **distinguishing power** of  $A$  (between  $X$  and  $Y$ ).

If  $\Delta_A(X, Y) \geq \epsilon > 0$ , then  $A$  is an  $\epsilon$ -**distinguisher**. If some such  $A$  exists, then  $X$

and  $Y$  are **distinguishable**.

Example:

$X$  = uniform distribution of  $x \in B^n$  with  $n$  even, Hamming-weight =  $\frac{n}{2}$  . // Hamming

weight  $hw(W)$  of  $W = w_1 w_2 \dots w_n \in \{0,1\}^n = \sum_1^n w_i$

$Y = U_n$

A : output :  $\begin{cases} 1 & \text{if } hw(x) = \frac{n}{2} \\ 0 & \text{Otherwise} \end{cases}$

$$\Delta_A(X, Y) = \color{red}{(1-0=1)} = 1 - E_A(U_n) \approx 1 - \frac{1}{\sqrt{\pi n/2}} \xrightarrow{n \rightarrow \infty} 1$$

An irritating approximative calculation led to this result. Because Prof. von zur Gathen wrote in circles and cleaned the whiteboard milliseconds after writing something, I could not copy all the text. And I believe it is a good idea to keep it out of my mind.

Definition:

A **bit-generator** is a function  $g: B^k \rightarrow B^{n(k)}$  with  $n(k) \gg k$  . Random variable  $g(U_k)$  on  $B^{n(k)}$  //maybe it's  $U_n$  , he wrote microscopic// hw-3-distinguisher  $A$  :

$$\Delta_A(f(U_3), U_6) = \frac{11}{16}$$

Another distinguisher  $B$  : predict  $y_4$  from  $y_1, y_2, y_3$  :  $y \in B^6$  :

$$B(y) = \begin{cases} 1 & \text{if } y_4 = \text{minority}(y_1, y_2, y_3) \\ 0 & \text{otherwise} \end{cases}$$

$$E(B(U_6)) = \frac{1}{2}$$

distinguishing power  $\Delta_A(f(U_3), U_6) = \frac{1}{8}$  . Family  $(g_k)_{k \in \mathbb{N}}$  of Boolean functions

$g_k: B^k \rightarrow B^{n(k)}$  with  $k \ll n(k)$

$y_0 \in g_k(B^k)$  , say  $y_0 = g_k(0 \dots 0)$  . Then  $\text{prob}_{g_k(U_k)}(y_0) \geq \frac{1}{2^k}$  and  $\text{prob}_{U_{n(k)}} = \frac{1}{2^{n(k)}}$  .

$$A(Y) = \begin{cases} 1 & \text{if } Y = y_0 \\ 0 & \text{otherwise} \end{cases} , \quad E(A(U_{n(k)})) = \frac{1}{2^{n(k)}} , \quad E(A(g_k(U_k))) \geq \frac{1}{2^k} \quad \text{suppose „="}$$

$$\Delta_A(g_k(U_k), U_{n(k)}) = \frac{1}{2^k} - \frac{1}{2^{n(k)}} \approx \frac{1}{2^k}$$

A function  $t: \mathbb{N} \rightarrow \mathbb{R}$  is **negligible**, if  $t(k) < \frac{1}{\text{prob}(k)} \forall$  polys.

$\forall d > 1$  there exists  $k_0$  so that  $t(k) < \frac{1}{k^d} \forall k \geq k_0$  .

Example:  $t = \frac{1}{2^k}$  ,  $t = \frac{1}{2^{(\log k)^2}} = \frac{1}{k^{\log k}}$

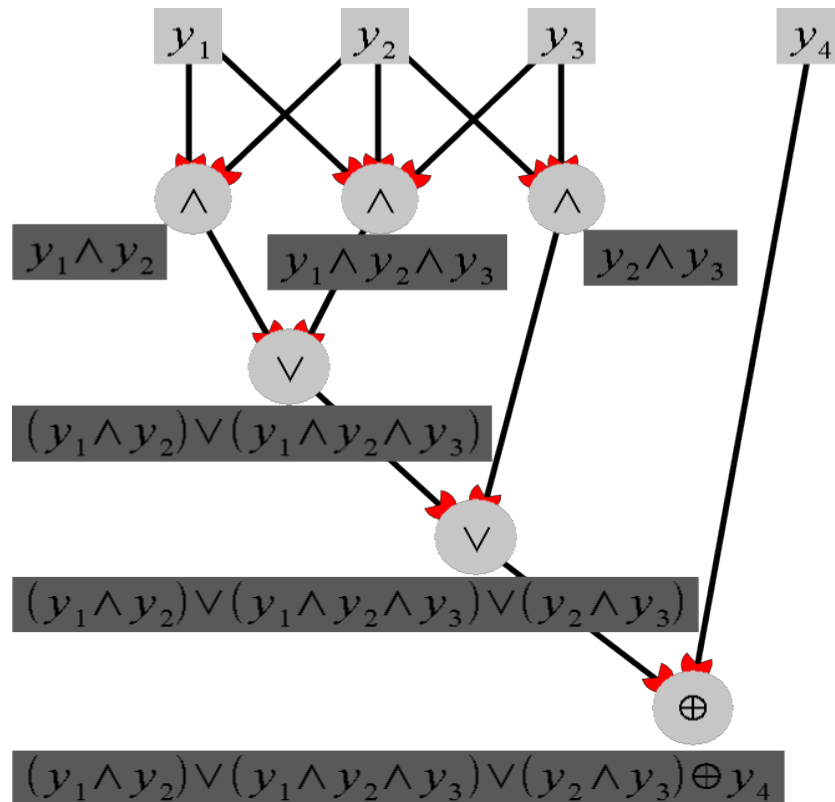
Definition: A family  $(g_k)_{k \in \mathbb{N}}$  is a pseudorandom-generator, if

- it can be computed in time  $\text{poly}(k)$
- for all probabilistic poly-time distinguishers  $A$  , the distinguishing power  $\Delta_A(g_k(U_k), U_{n(k)})$  is a negligible function of  $k$  .

**Boolean circuit**

inputs  $y_1, \dots, y_{n(k)}$  , operations  $\wedge, \vee, \neg, \oplus$

one output



$A := y_1 \wedge y_2$   
 $B := y_1 \wedge y_2 \wedge y_3$   
 $C := y_2 \wedge y_3$

$D := (y_1 \wedge y_2) \vee (y_1 \wedge y_2 \wedge y_3)$   
 $E := (y_1 \wedge y_2) \vee (y_1 \wedge y_2 \wedge y_3) \vee (y_2 \wedge y_3)$   
 $F := (y_1 \wedge y_2) \vee (y_1 \wedge y_2 \wedge y_3) \vee (y_2 \wedge y_3) \oplus y_4$

This circuit outputs „1“, if  $y_4$  belongs to the minority of  $\{y_1, y_2, y_3\}$  . The circuit can predict 50% of all  $y_4$  . By the way I can do that by the prediction-function 0 . I used the following table to proof the circuit's correctness. As I see it's correct in deed.

$y_1$	$y_2$	$y_3$	$y_4$	$A =$ $y_1 \wedge y_2$	$B =$ $y_1 \wedge y_2 \wedge y_3$	$C =$ $y_2 \wedge y_3$	$D =$ $A \vee B$	$E =$ $D \vee C$	$F =$ $E \neq y_4$
0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	1
0	1	0	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	1
0	1	1	0	0	0	1	0	1	1
0	1	1	1	0	0	1	0	1	0
1	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	1
1	0	1	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	1
1	1	0	0	1	0	0	1	1	1
1	1	0	1	1	0	0	1	1	0
1	1	1	0	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	0

lecture 28.05.09

Nach einer ausufernden Diskussion über Zufall beim DAX etc. ging's los mit dem Thema

### Pseudorandom generators

$$\Delta_A(X, Y) = (E(\underbrace{A(X)}_{g(U_k)}) - E(\underbrace{A(Y)}_{U_n = U_n(k)}))$$

$$\text{prob}(Y \leftarrow X) = \sum_{b \in B} \text{prob}(b \leftarrow Y) \cdot \text{prob}(b \leftarrow X)$$

i-th bit variable  $X_i(y)$  for  $y \in B^{i-1}$ ,  $X$  random variable with values in  $B$ .

for  $j \leq n$ ,  $w \in B^j$ :

$$p(w, *) := \text{prob}(w \leftarrow (X_1, \dots, X_j))$$

$$= \text{prob}(w \times U_{n-j} \leftarrow X) = \sum_{\substack{z \in B^{n-j} \\ // \text{ actually } z \in B^n \setminus B^j}} \text{prob}((w, z) \leftarrow X) \quad \text{for } b \in B$$

$$\text{prob}(b \leftarrow X_i(y)) = \frac{p(y, b, *)}{p(y, *)}, \text{ if } p(y, *) \neq 0, \text{ otherwise } \frac{1}{2}$$

Definition:

(i) A **predictor** for the  $i$ -th bit is a probabilistic algorithm with inputs from  $B^{i-1}$  and output in  $B$ .

Note: Every deterministic algorithm is a probabilistic algorithm.

(ii)  $X$  is a random variable on  $B^n$ ,  $(X_1, \dots, X_{i-1})$  the corresponding variable on  $B^{i-1}$ , is an  **$i$ -th bit predictor**. The **success rate**  $\sigma_P(X)$  of  $P$  is

$$\sigma_P(X) := \sum_{y \in B^{i-1}} p(y, x) \cdot \text{prob } P(P(y) \leftarrow X_i(y))$$

The **prediction power**  $P$  is

$$P = \sigma_P(X) - \frac{1}{2}.$$

if  $\sigma_P(X) \geq \frac{1}{2} + \epsilon$ , then  $P$  is an  **$\epsilon$ -predictor**.

(iii) A family  $(X_k)_{k \in \mathbb{N}}$  of random variables  $X_k$  on  $B^{n(k)}$ ,  $n(k) \gg k$ , is **unpredictable**, if any choice for the  $i_k$ -th bit of  $X_k$  has negligible prediction power.

Ja, wir haben heute tatsächlich 17 Zeilen geschafft!

lecture 01.06.09

**Pfingstferien**

lecture 05.06.09

**Pfingstferien**

lecture 08.06.09

The final exam is on August 12, 11-14h @ BitMax, if you fail, the second date is on September 15.

Allowed is a pocket calculator without computer algebra and a handwritten, two-sided DINA4-sheet with all information you can place on it. Copying is not allowed.

Pseudorandom generation

early 1980's: Manuel Blum, Micali, Goldwasser, Yao:

Theorem:

distinguisher

↔

predictor

$$A: B^n \rightarrow B$$

for  $x \in B^n$

$$P: B^{i-1} \rightarrow B$$

for  $1 \leq i \leq n$

generator  $g: B^k \rightarrow B^n$ ,  $k \ll n$

$X$  random variable on  $B^n$

$$X = g(U_k)$$

$$A(X) = P(X_1, \dots, X_{i-1}) \oplus X_i$$

$$E(A(X)) = E(P(\underbrace{X_1, \dots, X_{i-1}}_{\pi_{i-1}(X)}) \oplus X_i) = \sigma_P(X)$$

if  $\sigma_{(P)} \geq \frac{1}{2} + \epsilon$ , then  $\Delta_A(X, U_n) \geq \epsilon$

distinguisher  $\Rightarrow$  predictor

Theorem (Yao): Let  $A$  be an  $(\epsilon, s)$ -distinguisher (no idea what „s“ is...) between  $X$  and  $U_n$ . Then there exists an  $(\epsilon, s+2)$ -predictor for the  $i$ -th bit of  $X$ .

proof:

For  $0 \leq i \leq n$ , consider  $\pi_i: B^n \rightarrow B^i$  be the projection, and  $Y_i = \pi_i(X) \times U_{n-i}$  (That means  $Y_i$  consists of  $i$  bits from  $X$  and the rest is uniformly distributed) // note:  $Y_i$  hybrid,  $y_i$   $i$ -th bit of bitstring.

$Y_i$ : random variable on  $n$  bits, „first  $i$  according to  $X$ , last  $n-i$  uniform“ „hybrid distributions“

$$Y_0 = U_n, Y_n = X$$

$$e_i = E(A(Y_i))$$

We know:  $|e_0 - e_n| \geq \epsilon$ , assume  $e_0 - e_n \geq \epsilon$  (flip otherwise)

$$\epsilon \leq e_0 - e_n = \sum_{i \leq n} (e_{i-1} - e_i) \leq n \cdot \max(e_{i-1} - e_i)$$

$$\Rightarrow \max(e_{i-1} - e_i) \geq \frac{\epsilon}{n}$$

predictor  $P$ : input  $y \in B^{i-1}$ , output: 0 or 1.

1.  $(y_i, y_{i+1}, \dots, y_n) \leftarrow_R U_{n-i+1}$

2.  $z \leftarrow A(y, y_i, \dots, y_n)$

3. output  $z \oplus y_i$ .

Example:

$$A(X) = \begin{cases} 0 & \text{if Hammingweight } hw(X) = 3 \\ 1 & \text{else} \end{cases}$$

$$y \in B^6$$

$c_i(y) = \text{number of } (z_{i+1}, \dots, z_6)$

$y = (0\dots 0) \quad c_0(000000) = 20 \quad // \quad \binom{6}{3} \quad c_{-0} \Rightarrow \text{nothing fixed, } c_i \text{ first } i \text{ bits fixed: } \binom{n-i}{3}$

The example from projector:

		$c_0$	$c_1$	$c_2$	$c_3$	$c_4$	$c_5$	$c_6$
000	001101	20	10	4	3	2	1	1
001	001011	20	10	4	3	1	1	1
010	011010	20	10	6	3	2	1	1
011	010110	20	10	6	3	2	1	1
100	101100	20	10	6	3	1	1	1
101	100101	20	10	6	3	2	1	1
110	110100	20	10	4	3	1	1	1
111	110010	20	10	4	3	2	1	1

$i$	0	1	2	3	4	5	6
$e_i$	$\frac{5}{16}$	$\frac{5}{16}$	$\frac{5}{16}$	$\frac{3}{8}$	$\frac{13}{32}$	$\frac{1}{2}$	1
$e_i - e_{i-1}$	0	0	$\frac{2}{32}$	$\frac{1}{32}$	$\frac{3}{32}$	$\frac{16}{32}$	-

lecture 11.06.09

### Fronleichnam

lecture 15.06.09

Pseudorandom  $\Leftrightarrow$  unpredictable

From short to long generators

„  $\Leftarrow$  “

Predictor  $\hat{P} \leftarrow$  distinguisher  $\hat{A}$

hybrid distributions:  $Y_i = \pi(X) \times U_{n-i}$

$$e_i = E(\hat{A}(Y_i))$$

$$e_0 - e_1 \geq \epsilon$$

$$e_{i-1} - e_i \geq \frac{\epsilon}{n} \quad \text{for some } i$$

$\hat{P}$ : input  $y \in B^{i-1}$ , output 0 or 1

1. Choose  $y_i, \dots, y_n \in B$  uniformly at random.

2.  $y^* \leftarrow (y, y_i, \dots, y_n)$

3.  $z \leftarrow \hat{A}(y^*)$

Output  $y_i \oplus z \oplus 1$  // maybe to be switched

success rate of  $\hat{P}$  on  $X$  :

$$\begin{aligned}\sigma_{\hat{P}}(X) &= \sum_{y \in B^{i-1}} p(y, *) \cdot \text{prob}(\hat{P}(y) \leftarrow X_i(y)) \\ &= \sum_{y, y_i \in B^{i-1}} p(y, *) \cdot \text{prob}[(y_i \leftarrow U_i) \wedge (1 \oplus y_i \oplus \hat{A}((y, y_i) \times U_{n-i}) \leftarrow X(y))] \\ &= \sum_{y, y_i \in B^{i-1}} p(y, *) \cdot \text{prob}[(0 \leftarrow (\hat{A}((y, y_i) \times U_{n-i})) \wedge (y_i \leftarrow X_i(y)) \wedge (y_i \leftarrow U_i)] \\ &\quad + \text{prob}[(1 \leftarrow (\hat{A}((y, y_i) \times U_{n-i})) \wedge (y_i \leftarrow U_i)]\end{aligned}$$

Frohe Weihnachten

$$\begin{aligned}&= \sum_{y, y_i \in B^{i-1}} p(y, *) \cdot \text{prob}[(0 \leftarrow (\hat{A}((y, y_i) \times U_{n-i})) \wedge (y_i \leftarrow U_i)] \\ &\quad - \text{prob}[(0 \leftarrow (\hat{A}((y, y_i) \times U_{n-i})) \wedge (y_i \leftarrow U_i)] \\ &\quad + \text{prob}[(1 \leftarrow (\hat{A}((y, y_i) \times U_{n-i})) \wedge (y_i \leftarrow U_i) \wedge (y_i \leftarrow X_i(y))] \\ &= \frac{1}{2} \text{prob}[0 \leftarrow \hat{A}(y_{i-1})] + \text{prob}[0 \leftarrow \hat{A}(y_i)] - \frac{1}{2} \text{prob}[1 \leftarrow \hat{A}(y_i)] \\ &= \frac{1}{2} (1 - e_{i-1}) - (1 - e_i) + \frac{1}{2} e_i \\ &= -\frac{1}{2} - \frac{e_i - 1}{2} + \frac{3}{2} e_i = \frac{-1 - e_i - (-1) + 3e_i}{2} = \frac{2e_i}{2} = e_i \quad \text{Frohe Ostern}\end{aligned}$$

## Short to long integers

Suppose  $f: B^k \rightarrow B^{k+1}$  is pseudorandom,

We get a generator  $g: B^k \rightarrow B^{k+l}$  for  $l \geq 1$

Idea:  $f(x) = (y_1, \underbrace{y_2, \dots, y_{k+1}}_f) \rightarrow f(f(x)) = y_1, \underbrace{y_2, y_3, \dots, y_{k+2}}_{k+1}$  „keep applying the last k bits,“

$$\begin{aligned}f_i &:= \text{id}_B \times f: B^{i+k-1} \rightarrow B^{i+k} \\ (x_1, \dots, x_{i-1}, x_i, \dots, x_{i+k-1}) &\rightarrow (x_1, \dots, x_{i-1}, f(x_i, \dots, x_{i+k-1}))\end{aligned}$$

$$g_i := f_i \circ \dots \circ f_3 \circ f_2 \circ f_1$$

**Theorem:** let  $f: B^k \rightarrow B^{k+1}$  be an  $(\epsilon, s)$ -resilient generator that can be computed in size  $t$ ,  $l \geq 1$ .

$g = g_l: B^k \rightarrow B^{k+l}$ . Then  $g$  is an  $(l\epsilon, s - lt)$ -resilient generator, and can be computed with  $l$  applications of  $f$ .

**Proof:**  $h_i = \pi_1 \times g_{i-1}: B^{k+1} \rightarrow B^{k+i}$   
 $(x_1, \dots, x_{k+1}) \rightarrow (x_i, g_{i-1}(x_2, \dots, x_{k+1}))$

Then  $h_i \circ f = g_i$

$$h_i(f(U_k)) = g_i(U_k) \quad h_i(U_{k+1}) = U_i \times g_{i-1}(U_k)$$

Let  $\hat{A}$  be an  $(\delta, s')$ -distinguisher between  $g(U_k)$  and  $U_{k+l}$  :

$$E(\hat{A}(g(U_k))) - E(\hat{A}(U_{k+l})) \geq \delta$$

Distinguisher  $\hat{B}$  : input  $x \in B^{k+1}$  output: 0 or 1.

1. Choose  $i \in \{1, \dots, l\}$  uniformly at random

2.  $y \leftarrow U_{l-i}$

3. Execute  $\hat{A}$  on  $(y, h_i(x)) \in B^{k+l}$  and return its output.

$$\text{prob}(1 \leftarrow \hat{B}(x)) = \frac{1}{l} \sum_{i=1}^l \text{prob}(1 \leftarrow \hat{A}(U_{l-i}) \times h_i(x))$$

Hybrid variables  $Y_i = U_{l-i} \times g_i(U_k)$

$$Y_0 = U_l \times g_0(U_k) = U_{l+k},$$

$$Y_l = g_l(U_k)$$

$$Y_i = U_{l-i} \times g_i(U_k) = U_{l-i} \times h_i(f(U_k))$$

$$Y_{i-1} = U_{l-i+1} \times g_{i-1} = U_{l-i} \times U_i \times g_{i-1}(U_k) = U_{l-i} \times h_i(U_{k+1})$$

$$\alpha_i = \text{prob}(1 \leftarrow \hat{A}(Y_i)) \text{ for } 0 \leq i \leq l. \text{ Then } \alpha_l - \alpha_0 \geq \delta$$

$$\text{prob}(1 \leftarrow \hat{B}(f(U_k))) = \frac{1}{l} \sum_{i=1}^l \text{prob}(1 \leftarrow \hat{A}(U_{l-i}) \times h_i(f(U_k))) = \frac{1}{l} \sum_i \text{prob}(1 \leftarrow \hat{A}(Y_i)) = \frac{1}{l} \sum_{i=1}^l \alpha_i$$

$$\text{prob}(1 \leftarrow \hat{B}(U_{k+1})) = \frac{1}{l} \sum_{i=1}^l \text{prob}(1 \leftarrow \hat{A}(U_{l-i} \times h_i(U_{k+1}))) \stackrel{!}{=} \frac{1}{l} \sum_i \text{prob}(1 \leftarrow \hat{A}(Y_{i-1})) = \frac{1}{l} \sum_{i=1}^l \alpha_{i-1}$$

$$= E(\hat{B}(f(U_k))) - E(\hat{B}(U_{k+1}))$$

$$= \frac{1}{l} \left( \sum_{i=1}^l \alpha_i - \sum_{i=1}^l \alpha_{i-1} \right)$$

$$= \frac{1}{l} (\alpha_l - \alpha_0) \geq \frac{\delta}{l}$$

lecture 18.06.09

Repetition (answer to a question)

Yao: distinguisher  $\rightarrow$  predictor

$$g: B^k \rightarrow B^n \text{ output of } \hat{A} : b \in B$$

$$|E(\hat{A}(g(U_k))) - E(\hat{A}(U_k))| \leq \epsilon$$

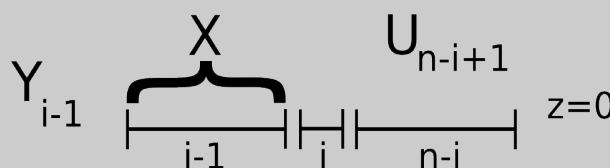
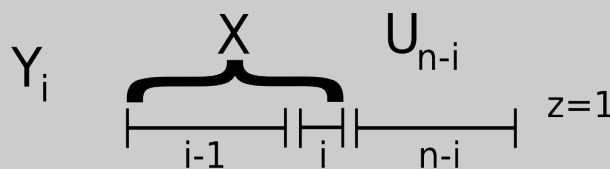
Consider k and n fixed, then we have circuits with a fixed size.

$$\hat{P}: B^{i-1} \rightarrow B \quad \hat{P} \text{ is actually not a function, because it can be probabilistic.}$$

There is a  $\hat{P}$ , which is an  $\epsilon$ -predictor.  $\Leftrightarrow$  There is an  $\hat{A}$ , which is an  $\epsilon$ -distinguisher.

$\frac{\epsilon}{n}$  at neighboring hybrid distribution

If we have a distinguisher,  $E(\hat{A}(g(U_k)))_{k \in B^i, B^{i-1}, \dots, B^1, B^0}$  with hybrid distributions



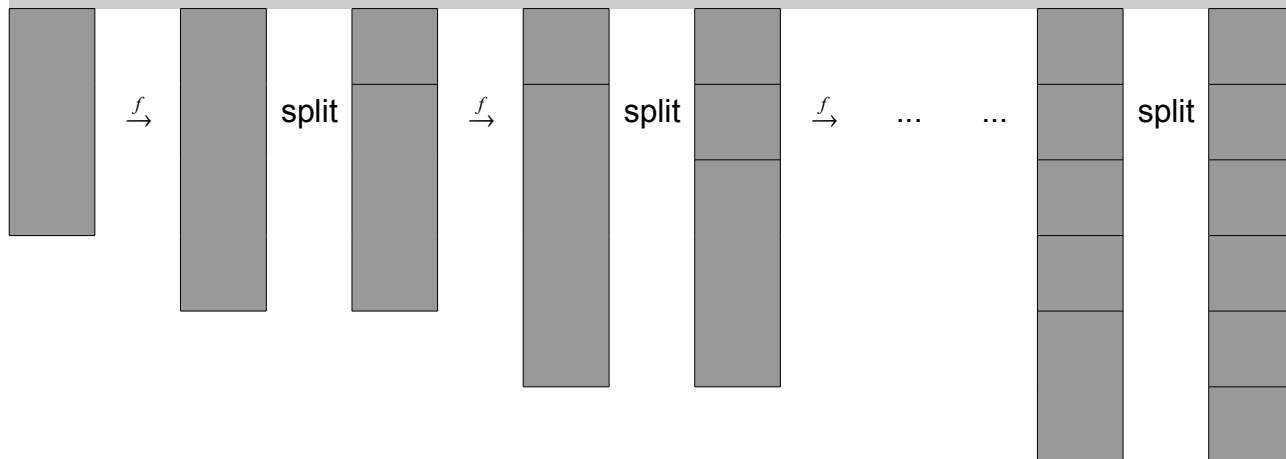
$$X = g(U_k)$$

probabilistic predictor  $\left( \overbrace{1 \dots i-1}^X \mid \overbrace{i \dots n}^U \right)$

artify n-bit input, output z  
compare i-th bit

short  $\rightarrow$  long

distinguishing power  $\geq \frac{\epsilon}{n}$



### The Nisan-Wigderson generator

Define when a function  $f: B^n \rightarrow B$  is „hard“. **( $\epsilon, s$ )-hard**: For all algorithms A with size s we

$$\text{have } \left| \text{prob} \left( \underbrace{(f \leftarrow \hat{A})(U_n)}_{=(f(x) \leftarrow \hat{A}(x)) \wedge (x \leftarrow U_n)} \right) - \frac{1}{2} \right| \leq \epsilon .$$

The **hardness** of  $f$   $H_f$  is the maximal integer  $h$  so that  $f$  is  $(h^{-1}, h)$ -hard.

Yao's XOR-Lemma:

Let  $f_1, \dots, f_k: B^k \rightarrow B$  all be  $(\epsilon, s)$ -hard,  $f = f_1 \oplus \dots \oplus f_k: B^{kn} \rightarrow B$  and a  $\delta > 0$ . Then  $f$  is  $(\epsilon^k + \delta, \delta^2(1-\epsilon)s)$ -hard.

**Nisan-Wigderson pseudorandom generators**

Philosophy: It's necessary and ok to assume  $P \neq NP$ , or the existence of some hard problem.

Boolean function:  $(\epsilon, s)$ -hard

integers  $k, n, s, t$ : A  $(k, n, s, t)$ -design is a sequence  $S_1, \dots, S_n$  of subsets  $\{1, \dots, k\}$  such that

- for all  $i, j \leq n: \#S = s$  ( $s$  just counts the elements of  $S$ ) and
- $\#(S_i \cap S_j) \leq t$ , if  $i \neq j$

7	8	9	◆	■	●	◆	■	●	■	◆	●	●	●
4	5	6	◆	■	●	●	◆	■	◆	●	■	■	■
1	2	3	◆	■	●	■	●	◆	●	■	◆	◆	◆

$n=12, k=9, s=3, t=1$

$S_1=\{1,4,7\}, S_5=\{1,6,8\}$

$$f_D \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} = 1001\dots$$

Theorem: Let  $k, n, s, \geq 1$  be integers,  $s \geq 2, t = \lfloor \log_2 n \rfloor - 1$ .

$f: B^s \rightarrow B$  with hardness  $H_f \geq 2n^2$ , and  $D$  a  $(k, n, s, t)$ -design. Then  $f_D := B^k \rightarrow B^n$  is an  $(n^{-1}, \dots)$

*He erased the whiteboard before I could read. But we proof what we don't know:*

Proof:

From any  $(n^{-1}, n)$ -distinguisher between  $f_D(U_k)$  and  $U_n$  we get an  $(n^{-2}, n)$ -predictor for the  $i$ -th bit of  $f_D(U_k)$ , for some  $i \leq n$ .

Why:  $S_i = \{1, \dots, s\} \subseteq \{1, \dots, k\}$  write  $U_k = U_s \times U_{k-s}$ .

$$\begin{aligned} \frac{1}{2} + \frac{1}{n^2} &\leq \sigma_P(f_D(U_k)) = \sum_{y \in B^{n-1}} \text{prob}(y \leftarrow X_1 \times \dots \times X_{i-1}) \cdot \text{prob}(\hat{P}(y) \leftarrow X_i(y)) \\ &= \sum_{x' \in B^s, x'' \in B^{k-s}} \text{prob}(x'' \leftarrow U_{k-s}) \cdot \text{prob}(x = f_D(U_k)) \cdot \text{prob}(f(x') \leftarrow \hat{P}(y)) \\ y = f_D(x', x'') \in B^{i-1} &= -2^{k-s} \sum_{x'' \in B^{k-s}} r(x'') // \text{something with average of } (k-s) \text{ elts.} \end{aligned}$$

where  $r(x'') = 2^{-s} \sum_{x' \in B^s} \text{prob}(f(x') \leftarrow \hat{P}(y))$   
 $y = f_D(x', x'')_{1 \dots i-1}$

The average of  $r$  is  $\geq \frac{1}{2} + \frac{1}{n^2}$ , so there exists some  $z \in B^{k-s}$  so that  $r(z) \geq \frac{1}{2} + \frac{1}{n^2}$ . //he then substituted  $x''$  by  $z$ .

Algorithm  $\hat{A}$  to compute  $f$  on input  $x' = (x_1, \dots, x_s) \in B^s$  :

1. for  $j=1, \dots, i-1$  do  $y_j \leftarrow f_D(x', z)_j$
2. Output  $\hat{P}(y)$ , where  $y = (y_1, \dots, y_{i-1})$

Success probability  $y = r(z) \geq \frac{1}{2} + \frac{1}{n^2}$

Cost of algorithm:

1.  $i-1$  evaluation of  $f$  // guess what he means ;)
2. evaluate  $\hat{P}(y)$

Why is 1. easy? For  $j < i$ ,  $\#(S_j \cap S_i) \leq t$  and  $f_D(x', s)_j$  depends only at most  $t$  bits in  $x'$ ; the others are fixed by  $z$ . Any Boolean function on  $t$  can be evaluated in size  $2^t$ .

lecture 25.06.09

***I missed class. But Felix allowed me to use his notes to fill this hole:***

### REPETITION: Nisan Wigderson (Wigderson?) generator

( $k, n, s, t$ )-design

We need a hard function + good design  $\rightarrow$  p.r.g.

A good design:  $F_q =$  finite field with  $q$  elements, so that  $q$  is a prime power. (Think:  $q$  is prime)

$$P = \{f \in F_q[x] : \text{deg } f \leq t\}, \text{ for } f \in P: \quad S_f = \underbrace{\{(u, f(u)) : u \in F_q\}}_{\text{Graph of } f} \subseteq F_q^2 = L$$

$$k = \#L = q^2$$

$$n = \#P = q^{t+1}$$

$$s = q$$

Why is the intersection of  $S_f$  and  $S_g \leq t$  ?

Let  $f, g \in P, f \neq g$

Suppose  $\#(S_f \cap S_g) \geq t+1$

Then  $f-g$  has  $\text{deg}(f-g) \leq t$  and  $\geq t+1$  roots.

$\Rightarrow f-g=0$  and therefore  $f=g$ .

$$k = 3^2 = 9 \quad n = 9 \quad s = 3 = q \quad t = 1$$

2	c	c	c
1	b	b	b
0	a	a	a
	0	1	2

f	e	d	
e	d	f	
d	f	e	
	0	1	2

i	g	h	
h	i	g	
g	h	i	
	0	1	2

$a: f=0$	$d: x$	$g: f=2x$
$b: f=1$	$e: x+1$	$h: 2x+1$
$c: f=2$	$f: x+1$	$i: 2x+2$

Result:

- (i) For  $s > t$ ,  $s$  a prime power,  $\exists(k, n, s, t)$  -design with  $h = f^2$ ,  $s^{t+1}$ ,  $s, t \rightarrow (s^2, s^{t+1}, s, t)$  design.
- (ii) For  $k, n, t$  and a prime power  $s$  with  $k \geq s^2$ ,  $t \geq (\log_s n) - 1$ , there exists a  $(k, n, s, t)$ -design.

**Nisan-Wigderson theorem:**

Let  $s$  be a prime power and  $f: B^s \rightarrow B$  with hardness  $H_f \geq 2n^\square$

Then the generator  $f_D$  is a pseudorandom generator from  $B^k \rightarrow B^n$ , where  $D$  is the design from (ii).

Counting circuits of Boolean functions  $\Rightarrow$   
 almost every  $f: B^s \rightarrow B$  requires circuits of  
 size  $\geq \frac{2^s}{s}$

Idea: „Existence of p-r-g-s-  $\Rightarrow$  Existence of hard functions“  
 $\Leftarrow$   
 Nisan-Wigderson

Asymptotic view of the world:

Some complexity classes:

$$P \subseteq BPP \subseteq NP \subseteq EXP$$

$g = (g_k)_{k \in \mathbb{N}}$  with  $g_k: B^k \rightarrow B^{n(k)}$  is **quich**, if it can be implemented, if it can be in time exponential in  $k$ .

$r: \mathbb{N} \rightarrow \mathbb{N}$  is reasonable, if for  $k, k' \in \mathbb{N}$ :

$$k \leq r(k) \leq 2^k$$

$$r(k) \leq r(k') \text{ if } k \leq k'$$

$$(r(k))^2 \leq r(k^2)$$

**Theorem:**

Let  $r: \mathbb{N} \rightarrow \mathbb{N}$  be reasonable. Then the following facts are equivalent:

- (i) For some  $c > 0 \exists f \in EXP$  with hardness  $r(k^c)$
- (ii) For some  $c > 0 \exists$  quich prg  $g$  with  $g_k: B^k \rightarrow B^{r(k^c)}$

Usual conjecture:  $P \subseteq BPP \rightarrow$  derandomization

PI: Polynomial Identity testing  $\in BPP$   
 Impagliazzo & Kabanets:  $P = BPP \Leftrightarrow PI \in P$

prg:  
 distinguisher  
 predictor  
 short  $\rightarrow$  long  
 Nisan Wigderson

### Blum-Blum-Shub

$N = p \cdot q$  ,  $p \neq q$  odd primes,  $x_0 \in \mathbb{Z}_N^x$  random for  $i=1,2,\dots$   $x_i = x_{i-1}^2 \in \mathbb{Z}_N^x$  .

Output: Least significant bit (LSB) of  $x_0, x_1, \dots$

$$\sqrt{1} \qquad x^2 \qquad a^2=1 \qquad a \leq \pm 1$$

$$N | (a^2 - 1) = (a-1)(a+1)$$

$\Rightarrow \gcd(N, a-1)$  is a nontrivial factor of  $N$  .

### Lemma:

**Factoring**  $N \leq_{\text{PolynomialTimeReduction}}$  **square roots mod  $N$**  . (If I can determine square roots mod  $N$  in time  $O(f(n))$  , then I can also determine factors in  $O(f^c(n))_{c \in \mathbb{N}}$  .)

### Proof:

1. Pick  $x \in \mathbb{Z}_N^x$  uniformly at random,  $y = x^2 \in \mathbb{Z}_N^x$  .
2. Call square root routine with input  $y$  and output  $z$  .
3. Return  $g := \gcd(N, x-z)$ , if  $g \in \{1, N\}$  //  $g = 1$  or  $g = N$

$$\text{TODO: } \text{prob}(g \notin \{1, N\}) = \frac{1}{2}$$

$$\text{if } N = p \cdot q \text{ , } a = \frac{x}{z} \text{ , } a^2 = \frac{x^2}{z^2} = \frac{y}{y} = 1 \text{ .}$$

$\text{prob}(a \pm 1) = \frac{1}{2}$  , because  $\text{prob}(x = \pm z) = \frac{1}{2}$  , since 4 values are equally likely.

### Factoring assumption:

Factoring integers is hard (not poly-time)

### Square root assumption:

Taking square roots mod composite integers is hard.

Fact: Square roots modulo primes are easy.

square  $a = b^2 \in \mathbb{Z}_N^x$  .

Fermat (Little Theorem):  $1 = b^{p-1} = b^{\frac{2 \cdot p-1}{2}} = a^{\frac{p-1}{2}}$

How many squares?  $x^{\frac{p-1}{2}} - 1 = \prod_{a \in \mathbb{Z}_N^*, a \text{ square}} (x-a)$

$\forall a: a | x^{\frac{p-1}{2}} - 1$  : the (x-a)'s are coprime, so  $\prod (x-a) | x^{\frac{p-1}{2}} - 1$

Fermat polynomial  $= x^p - x = \prod_{a=0} x \cdot \prod_{a \text{ square}} (x^{\frac{p-1}{2}} - 1) \prod_{a \text{ nonsquare}} (x^{\frac{p-1}{2}} + 1) = \prod_{a \in \mathbb{Z}_N} (x-a)$  //NOTE: not  $\mathbb{Z}_N^x$  !

$$x^{p-1} - 1 = \prod_{a \in \mathbb{Z}_N^*} (x-a)$$

This leads to random polynomial time factorization of polynomials in  $\mathbb{Z}_p(x)$  of the form  $\prod_{a \in A} (x-a)$  for some  $A \subseteq \mathbb{Z}_p$ .

Deterministic: ??

$$\underbrace{0}_{LS} \cdot \underbrace{x}_{LS} \cdot \underbrace{(x^{\frac{p-1}{2}} - 1)}_{LS} \underbrace{(x^{\frac{p-1}{2}} - (-1))}_{LS} //LS = \text{Legendre Symbol}$$

Legendre ~1785

**Legendre Symbol:**

prime  $p$ ,  $a \in \mathbb{Z}_p$

$$\left(\frac{a}{p}\right) = a^{\frac{p-1}{2}} = \begin{cases} 1 & , \text{if } a \text{ square unit} \\ -1 & , \text{if } a \text{ nonsquare unit} \\ 0 & , \text{if } a = 0 \end{cases}$$

**Jacobi Symbol:**

$$N = p \cdot q$$

$$\left(\frac{a}{N}\right) = \left(\frac{a}{p}\right) \cdot \left(\frac{a}{q}\right)$$

$$\exists b: a = b^2 \text{ mod } N \Leftrightarrow \exists b_1, b_2: a = b_1^2 \text{ mod } p \wedge a = b_2^2 \text{ mod } q$$

conditions →  
↓

$$\frac{a}{p} = 1$$

$$\frac{a}{q} = -1$$

$$\frac{a}{q} = 1$$

$$\frac{a}{p} = -1$$

$\frac{a}{N} = 1$ $\square_N$ $\square_p \times \square_q$ $(u, -v)$	$\frac{a}{N} = -1$ $\square_p \times X \square_q$ $(u, v)$
$\frac{a}{N} = -1$ $X \square_p \times \square_q$ $(-u, -v)$	$\frac{a}{N} = 1$ $X \square_N$ $X \square_p \times X \square_q$ $(-u, v)$

generator will concentrate on diagonal boxes.

Jacobi Symbol  $\left(\frac{a}{N}\right)$  can be computed in polynomial time.