

Stefan Sokołowski

MATEMATYKA DYSKRETNA

wykład 7

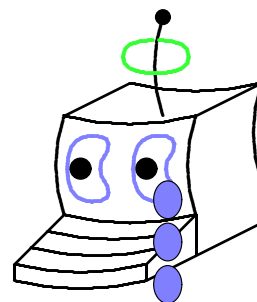
Inst. Informatyki Stosowanej, ANS
Elbląg, 2024/2025

Wykład 7, str. 1

Nieprawdziwa arytmetyka komputerowa

Zbiór $\mathbb{Z}$ liczb całkowitych jest **nieskończony**
— więc w komputerze się nie zmieści. . .

Nie wszystkie obliczenia będą więc poprawnie wykonalne. Istotne jest wiedzieć, kiedy nawalają i jakie mogą być tego konsekwencje.



Możliwe rozwiązania, kiedy nie da się dostarczyć poprawnego wyniku:

- (A) dostarczać **nie**poprawny wynik
- (B) sygnalizować niemożność

Dla obliczeń na liczbach całkowitych na ogół stosuje się wersję (A). . .

Co dzieje się w komputerze z dużymi całkowitymi?

„Ludzki” system liczenia:

mnożenie przez **10** — przesunąć liczbę w lewo i zwolnione pole z prawej wypełnić zerem.

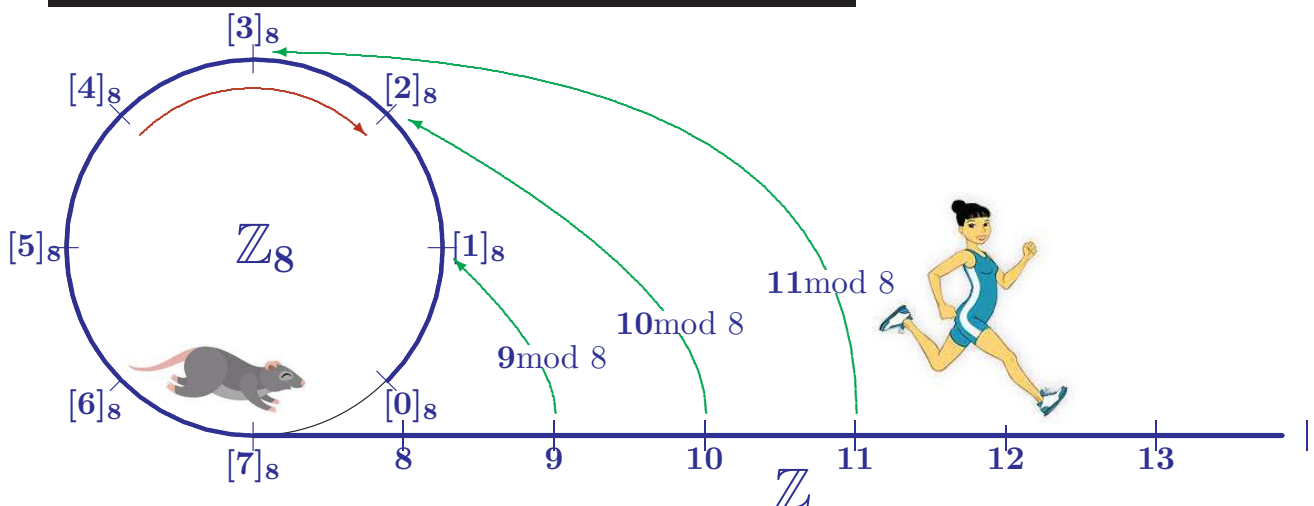
„Komputerowy” system liczenia:

mnożenie przez **2** — przesunąć liczbę w lewo i zwolnione pole z prawej wypełnić zerem.

[illegible]
$$\text{LICZBA} \times 2^{\text{długość pola}} = 0 \quad \text{czyli} \quad 2^{\text{długość pola}} \text{ jest jak } 0 \dots$$

...całkiem jak w $\mathbb{Z}_2$ długość pola

Reszty z dzielenia biegną „w kółko”



$\mathbb{Z}_8$ — klasy liczb o tej samej reszcie:

$$[0]_8 = \{\dots, -16, -8, 0, 8, 16, \dots\}$$

$$[1]_8 = \{\dots, -15, -7, 1, 9, 17, \dots\}$$

.....

$$[7]_8 = \{\dots, -9, -1, 7, 15, 23, \dots\}$$

$$[1]_8 = [9]_8 = [-7]_8 = \dots$$

Działania w $\mathbb{Z}_8$

Przekształcenie $[\dots]_8 : \mathbb{Z} \rightarrow \mathbb{Z}_8$ wprowadza działania na $\mathbb{Z}_8$:

$$\begin{aligned} [a]_8 + [b]_8 &= [a + b]_8 \\ [a]_8 - [b]_8 &= [a - b]_8 \\ [a]_8 \cdot [b]_8 &= [a \cdot b]_8 \end{aligned}$$

Przykład:

$$[7]_8 + [6]_8 = [13]_8 = [5]_8$$

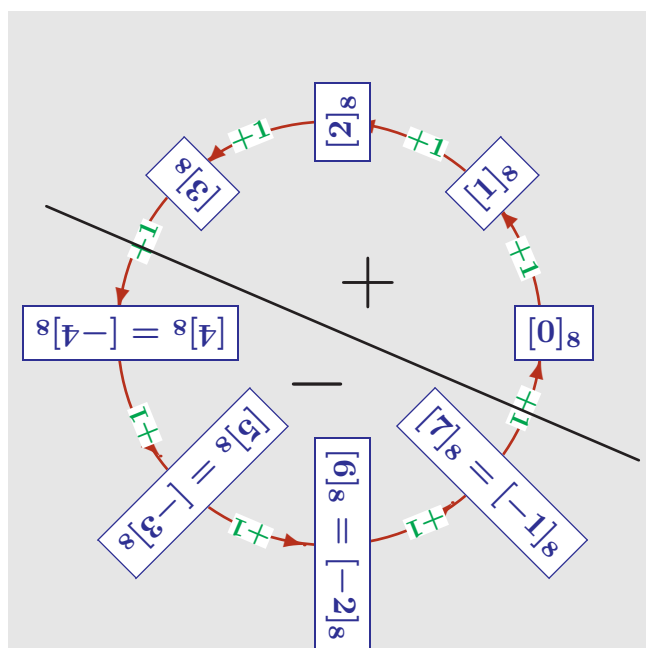
$$[3]_8 - [5]_8 = [-2]_8 = [6]_8$$

$$[5]_8 \cdot [7]_8 = [35]_8 = [3]_8$$

$$\begin{aligned} [35]_8 &= [35]_8 - [4]_8 \cdot [0]_8 \\ &= [35]_8 - [4]_8 \cdot [8]_8 \\ &= [35]_8 - [4 \cdot 8]_8 \\ &= [35 - 32]_8 \\ &= [3]_8 \end{aligned}$$

Komputerowy int

Gdyby na `int` przeznaczać po 3 bity, to typ `int` zawierałby $2^3 = 8$ różnych liczb. Arytmetyka takich liczb: $\mathbb{Z}_{2^3}$.



$[0]_8$	$=$	$\boxed{000}$	
$[1]_8$	$=$	$\boxed{001}$	
$[2]_8$	$=$	$\boxed{010}$	
$[3]_8$	$=$	$\boxed{011}$	
$[-4]_8$	$=$	$\boxed{100}$	} ← ujemne
$[-3]_8$	$=$	$\boxed{101}$	
$[-2]_8$	$=$	$\boxed{110}$	
$[-1]_8$	$=$	$\boxed{111}$	

Komputerowy int

Naprawdę na `int` przeznaczają się (zwykle) 4 bajty czyli 32 bity. Typ `int` zawiera $2^{32} \approx 4$ mld. liczb. Arytmetyka komputerowa: $\mathbb{Z}_{2^{32}}$.

Póki liczymy na małych liczbach — wszystko jak w $\mathbb{Z}$.

W okolicach 2 mld. — należy uważać.

```
#include<stdio.h>

int main() {
    int a = 65536, b = 65535;
    printf(" %d * %d = %d\n", a, b, a*b);
    return 0;
}
```

Ten program mnoży przez siebie dwie liczby dodatnie i drukuje:

$$65536 * 65535 = -65536$$

(proszę osobiście sprawdzić).

Dlaczego?

$$65536 = 2^{16} \quad 65535 = 2^{16} - 1$$

$$\begin{aligned} [65536]_{2^{32}} \cdot [65535]_{2^{32}} &= [2^{16}]_{2^{32}} \cdot ([2^{16} - 1]_{2^{32}}) \\ &= [2^{16}]_{2^{32}} \cdot [2^{16}]_{2^{32}} - [2^{16}]_{2^{32}} \cdot 1 = [2^{32}]_{2^{32}} - [2^{16}]_{2^{32}} \\ &= [0]_{2^{32}} - [65536]_{2^{32}} = [-65536]_{2^{32}} \end{aligned}$$

Komputerowy int

```
#include<stdio.h>

int main() {
    int n=0, sil=1;
    while (n < 20) {
        printf(" %2i! == %10i\n", n, sil);
        n++; sil*=n;
    }
    return 0;
}
```

```
0! == 1
1! == 1
2! == 2
3! == 6
4! == 24
5! == 120
6! == 720
7! == 5040
8! == 40320
9! == 362880
10! == 3628800
11! == 39916800
12! == 479001600
13! == 1932053504
14! == 1278945280
15! == 2004310016
16! == 2004189184
17! == -288522240
18! == -898433024
19! == 109641728
```

$1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot \dots \cdot 13$
 $\swarrow \quad \searrow$
 $\quad \quad = 10$
13! powinno kończyć się zerem!

Komputerowy long

int — 32 bity — ok. 4 miliardów ($4 \cdot 10^9$) różnych liczb

long — 64 bity — ok. 16 trylionów ($16 \cdot 10^{18}$) różnych liczb

```
#include<stdio.h>

int main() {
    int n=0; long sil=1;
    while (n < 22) {
        printf(" %2i! == %20li\n", n, sil);
        n++; sil*=n;
    }
    return 0;
}
```

```
.....
10! ==          3628800
11! ==          39916800
12! ==          479001600
13! ==          6227020800
14! ==          87178291200
15! ==          1307674368000
16! ==          20922789888000
17! ==          355687428096000
18! ==          6402373705728000
19! ==          121645100408832000
20! ==          2432902008176640000
21! ==         -4249290049419214848
```

ujemna silnia?! 

Przez zwiększenie liczby bitów z 32 do 64

- poprawiliśmy dopuszczalną wielkość liczb z 4 mld do **aż** 16 trln
- poprawiliśmy bezbłędne obliczanie silni z 12! do **tylko** 20!

Propozycja „chińska”

Czy obliczenia na dużych liczbach całkowitych można zastąpić obliczeniami na mniejszych liczbach?

Stosując komputerowy int, obliczenia w $\mathbb{Z}$ zastępujemy obliczeniami w $\mathbb{Z}_{2^{32}}$.

Stosując chińskie twierdzenie o resztach, obliczenia w $\mathbb{Z}$ można by zastąpić np. przez obliczenia osobno w $\mathbb{Z}_9$, $\mathbb{Z}_{25}$ i $\mathbb{Z}_7$ (9, 25 i 7 są parami wzgl. pierwsze).

Wniosek: (z chińskiego o resztach)

Każdą liczbę $n \in \mathbb{Z}$ będziemy reprezentować przez trójkę

$(n \bmod 9, n \bmod 25, n \bmod 7)$

Wymagane w twierdzeniu współczynniki s_1, s_2, s_3 mają spełniać zależności

$$25 \cdot 7 \cdot s_1 \equiv_9 1$$

$$9 \cdot 7 \cdot s_2 \equiv_{25} 1$$

$$9 \cdot 25 \cdot s_3 \equiv_7 1$$

rozwiązanie:

$$s_1 = -2$$

$$s_2 = 2$$

$$s_3 = 1$$

Wtedy trójka (a_1, a_2, a_3) reprezentuje liczbę

$$n = (25 \cdot 7 \cdot (-2) \cdot a_1 + 9 \cdot 7 \cdot 2 \cdot a_2 + 9 \cdot 25 \cdot 1 \cdot a_3) \bmod (9 \cdot 25 \cdot 7)$$

czyli

$$n = (-350 \cdot a_1 + 126 \cdot a_2 + 225 \cdot a_3) \bmod 1575$$

Propozycja „chińska”

Przykład:

Jak policzyć $48 \cdot 21 + 37$?

$$\begin{array}{l}
 \mathbb{Z} \longrightarrow (\mathbb{Z}_9, \mathbb{Z}_{25}, \mathbb{Z}_7) \\
 48 \longrightarrow (3, -2, -1) \\
 21 \longrightarrow (3, -4, 0) \\
 37 \longrightarrow (1, 12, 2) \\
 \hline
 (1, -5, 2) \longrightarrow (-350 \cdot 1 + 126 \cdot (-5) + 225 \cdot 2) \bmod 1575 = 1008
 \end{array}$$

Co na tym zyskujemy/tracimy?

- główne obliczenia w $\mathbb{Z}_9$, $\mathbb{Z}_{25}$ i $\mathbb{Z}_7$ są prostsze niż w $\mathbb{Z}$
- ale potrzebne jest jednorazowe wyliczenie trójek i jednorazowy powrót do $\mathbb{Z}$
- to oczywiście nie muszą być trójki — im więcej niewielkich modułów tym prostsze obliczenia
- metoda jak skrojona dla obliczeń współbieżnych, z użyciem osobnych procesorów do różnych modułów

Funkcja φ Eulera

Definicja:

$$\begin{aligned}
 \varphi : \{n \in \mathbb{N} \mid n > 0\} &\rightarrow \mathbb{N} \\
 \varphi(n) &\stackrel{\text{def}}{=} \left| \{k \in \mathbb{N} \mid 1 \leq k \leq n \wedge \text{nwd}(k, n) = 1\} \right|
 \end{aligned}$$

Czyli słowami:

— $\varphi(n)$ określa, ile w przedziale $[1..n]$ jest liczb względnie pierwszych z n .

Przykład:

Dla $n = 12$ wzgl. pierwsze są 1, 5, 7, 11,
więc $\varphi(12) = 4$.



Leonhard Euler
1707–1783

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	...
$\varphi(n)$	1	1	2	2	4	2	6	4	6	4	10	4	12	6	8	...

Funkcja φ Eulera

Własność funkcji φ :

n jest liczbą pierwszą $\iff \varphi(n) = n - 1$

Dowód:

$\Rightarrow$:

Jeśli n jest liczbą pierwszą, to jest wzgl. pierwsza z każdą z liczb z przedziału $[1..n-1]$, których jest $n-1$.

$\Leftarrow$:

Niech $\varphi(n) = n - 1$ i $n = p \cdot q$, $1 < p, q < n$.

Wtedy $\text{nwd}(p, n) = \text{nwd}(p, pq) = p$,

więc p **nie jest** wzgl. pierwsze z n ,

więc zbiór liczb w $[1..n]$ wzgl. pierwszych z n jest mniejszy niż $n-1$,

więc $\varphi(n) < n-1$, sprzeczność z tym.

□

Funkcja φ Eulera

Własność funkcji φ :

Jeśli n jest liczbą pierwszą, to

$\varphi(n^k) = n^{k-1} \cdot (n - 1)$ dla dowolnego $k \in \mathbb{N}$, $k > 0$.

Dowód:

Skoro n jest liczbą pierwszą, to n^k dzieli się tylko przez potęgi n ; wobec tego ma wspólny dzielnik **większy niż 1** tylko z wielokrotnościami n , czyli z liczbami ze zbioru

$$\{1n, 2n, 3n, \dots, n^{(k-1)}n\} \quad (*)$$

Z pozostałymi liczbami przedziału $[1..n^k]$ liczba n^k jest wzgl. pierwsza. Liczb względnie pierwszych z n^k jest więc w tym przedziale

$$\begin{aligned} \varphi(n^k) &= (\text{liczba elem. przedz. } [1..n^k]) - (\text{liczba elem. zb. } (*)) \\ &= n^k - n^{k-1} = n^{k-1} \cdot (n - 1) \end{aligned}$$

□

Funkcja φ Eulera

Własność funkcji φ :

Jeśli $m, n > 0$ i $\text{nwd}(m, n) = 1$, to $\varphi(mn) = \varphi(m) \cdot \varphi(n)$

Przykład:

Policzyć $\varphi(2025)$.

$$\begin{aligned}\varphi(2025) &= \varphi(25 \cdot 81) = \varphi(5^2 \cdot 3^4) && (5^2 \text{ i } 3^4 \text{ są wzgl. pierwsze}) \\ &= \varphi(5^2) \cdot \varphi(3^4) \\ &= 5^1 \cdot (5 - 1) \cdot 3^3 \cdot (3 - 1) \\ &= 20 \cdot 54 = 1080\end{aligned}$$

Przykład:

Policzyć $\varphi(710\,875)$.

$$\begin{aligned}\varphi(710\,875) &= \varphi(5^3 \cdot 11^2 \cdot 47) && (5^3, 11^2 \text{ i } 47 \text{ są wzgl. pierwsze}) \\ &= \varphi(5^3) \cdot \varphi(11^2) \cdot \varphi(47) \\ &= 5^2 \cdot 4 \cdot 11 \cdot 10 \cdot 46 \\ &= 506\,000\end{aligned}$$

Funkcja φ Eulera

Własność funkcji φ :

$$\text{Jeśli } \left\{ \begin{array}{l} n = p_1^{a_1} p_2^{a_2} \dots p_k^{a_k}, \text{ gdzie} \\ p_1, p_2, \dots, p_k \text{ są liczbami pierwszymi} \\ a_1, a_2, \dots, a_k \in \mathbb{N} \text{ są dodatnie} \end{array} \right\} \text{ to } \varphi(n) = n \cdot \prod_{i=1}^k \left(1 - \frac{1}{p_i}\right)$$

Dowód:

$$\begin{aligned}\varphi(n) &= \varphi(p_1^{a_1} p_2^{a_2} \dots p_k^{a_k}) = \varphi(p_1^{a_1}) \cdot \varphi(p_2^{a_2}) \cdot \dots \cdot \varphi(p_k^{a_k}) \\ &= p_1^{a_1} \left(1 - \frac{1}{p_1}\right) \cdot p_2^{a_2} \left(1 - \frac{1}{p_2}\right) \cdot \dots \cdot p_k^{a_k} \left(1 - \frac{1}{p_k}\right) \\ &= \underbrace{p_1^{a_1} p_2^{a_2} \dots p_k^{a_k}}_n \cdot \left(1 - \frac{1}{p_1}\right) \cdot \left(1 - \frac{1}{p_2}\right) \cdot \dots \cdot \left(1 - \frac{1}{p_k}\right) \\ &= n \cdot \prod_{i=1}^k \left(1 - \frac{1}{p_i}\right)\end{aligned}$$

□

Funkcja φ Eulera

Przykład:

Policzyć $\varphi(2025)$.

$$\begin{aligned}\varphi(2025) &= \varphi(5^2 \cdot 3^4) = 2025 \cdot \left(1 - \frac{1}{5}\right) \cdot \left(1 - \frac{1}{3}\right) \\ &= 2025 \cdot \frac{4}{5} \cdot \frac{2}{3} = \frac{2025 \cdot 4 \cdot 2}{5 \cdot 3} = 1080\end{aligned}$$

Przykład:

Policzyć $\varphi(710\,875)$.

$$\begin{aligned}\varphi(710\,875) &= \varphi(5^3 \cdot 11^2 \cdot 47) \\ &= 710\,875 \cdot \left(1 - \frac{1}{5}\right) \cdot \left(1 - \frac{1}{11}\right) \cdot \left(1 - \frac{1}{47}\right) \\ &= 710\,875 \cdot \frac{4}{5} \cdot \frac{10}{11} \cdot \frac{46}{47} = 506\,000\end{aligned}$$

Funkcja φ Eulera

Twierdzenie Eulera:

Jeśli $\text{nwd}(a, n) = 1$ i $n \geq 2$, to $a^{\varphi(n)} \equiv_n 1$

Dowód:

Niech $P = \{p_1, p_2, \dots, p_{\varphi(n)}\}$

— zbiór liczb wzgl. pierwszych z n z przedziału $[1 \dots n]$.

Niech $r_i = (ap_i) \bmod n$; wtedy $\text{nwd}(r_i, n) = \text{nwd}(ap_i, n) = 1$, więc $r_i \in P$.

Wszystkie r_i są różne, bo $ap_i \equiv_n ap_j \Rightarrow p_i \equiv_n p_j$. Więc $\{r_1, r_2, \dots, r_{\varphi(n)}\} = P$.

Dlatego

$$(ap_1)(ap_2) \dots (ap_{\varphi(n)}) \equiv_n p_1 p_2 \dots p_{\varphi(n)}$$

Kongruencję można podzielić obustronnie przez $p_1 p_2 \dots p_{\varphi(n)}$, bo ten iloczyn jest wzgl. pierwszy z n :

$$a^{\varphi(n)} \equiv_n 1$$

□

Funkcja φ Eulera

Przykład:

Policzyć $3^{1\,000\,000} \bmod 76$.

$$\begin{aligned}\varphi(76) &= \varphi(2^2 \cdot 19) = 2^2 \cdot 19 \cdot \left(1 - \frac{1}{2}\right) \cdot \left(1 - \frac{1}{19}\right) \\ &= \frac{2^2 \cdot 19 \cdot 1 \cdot 18}{2 \cdot 19} = 2 \cdot 1 \cdot 18 = 36\end{aligned}$$

Dlatego $3^{36} \equiv_{76} 1$.

$$\begin{aligned}3^{1\,000\,000} &= 3^{36 \cdot 27\,777 + 28} = 3^{36 \cdot 27\,777} \cdot 3^{28} = (3^{36})^{27\,777} \cdot 3^{28} \\ &\equiv_{76} 1^{27\,777} \cdot 3^{28} = 3^{28} = 3^{7 \cdot 4} = 2187^4 \equiv_{76} 59^4 \\ &= 3481^2 \equiv_{76} 61^2 = 3721 \equiv_{76} 73\end{aligned}$$

redukcja wykładnika
z miliona do 28

Funkcja φ Eulera

Przykład:

Policzyć $7^{1\,234\,567} \bmod 100$.

$$\begin{aligned}\varphi(100) &= \varphi(2^2 \cdot 5^2) = 2^2 \cdot 5^2 \cdot \left(1 - \frac{1}{2}\right) \cdot \left(1 - \frac{1}{5}\right) \\ &= \frac{2^2 \cdot 5^2 \cdot 1 \cdot 4}{2 \cdot 5} = 2 \cdot 5 \cdot 1 \cdot 4 = 40\end{aligned}$$

Dlatego $7^{40} \equiv_{100} 1$.

$$\begin{aligned}7^{1\,234\,567} &= 7^{40 \cdot 30\,864 + 7} = (7^{40})^{30\,864} \cdot 7^7 \\ &\equiv_{100} 1^{30\,864} \cdot 7^{3 \cdot 2} \cdot 7 = 343^2 \cdot 7 \equiv_{100} 43^2 \cdot 7 \\ &= 1849 \cdot 7 \equiv_{100} 49 \cdot 7 = 343 \equiv_{100} 43\end{aligned}$$

redukcja wykładnika
z 1 234 567 do 7

Funkcja φ Eulera

Po co nam funkcja Eulera?

- ułatwia (umożliwia) liczenie reszt z wysokich potęg:

$$a^b \bmod n$$

- jak to robić?

– policzyć resztę z wykładnika potęgi przez wartość funkcji Eulera na module:

$$r = b \bmod \varphi(n)$$

r jest na ogół **znacznie mniejsze** niż b

– z tw. Eulera wynika, że

$$a^b \bmod n = a^r \bmod n$$

- takie reszty z potęg potrzebne są np. **w szyfrowaniu**.

Szyfrowanie symetryczne

— odszyfrowywanie jest dokładną odwrotnością szyfrowywania;
znając sposób szyfrowania, potrafimy również odszyfrowywać

Przykład: Szyfr Cezara

zamiast	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
piszemy	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E

ALA MA KOTA $\longrightarrow$ FQF RF PTYF

Klucz:

wielkość przesunięcia; tutaj 5

Siła:

banalnie łatwy do złamania

Szyfrowanie symetryczne

— odszyfrowywanie jest dokładną odwrotnością zaszyfrowywania;
znając sposób szyfrowania, potrafimy również odszyfrowywać

Przykład: Prosty szyfr przestawieniowy

ALA MA KOTA $\rightarrow$ A A M O A T K A L
 1 3 5 7 9 11 10 8 6 4 2

Klucz:

sposób przestawiania

Siła:

zależna od sposobu przestawiania; na ogół niezbyt trudny

Szyfrowanie symetryczne

Przykład: Dodawanie długiego klucza

klucz		tekst jawny		zaszyfrowany	
L	11	A	0	11	L
I	8	L	11	19	T
T	19	A	0	19	T
W	22	spacja	26	21	V
O	14	M	12	26	spacja
spacja	26	A	0	26	spacja
O	14	spacja	26	13	N
J	9	K	10	19	T
C	2	O	14	16	Q
Z	25	T	19	17	R
Y	24	A	0	24	Y

Klucz:

dowolny tekst; np. jakaś powieść

Siła:

ze znajomością klucza — złamanie banalnie łatwe

bez znajomości klucza — bardzo trudne

Szyfrowanie **asymetryczne**

Dwa osobne klucze:

- do **zaszyfrowywania**, oraz
- do **odszyfrowywania**,

takie, że ze znajomości jednego nie wynika znajomość drugiego (albo przynajmniej, że to odwrócenie jest trudne do realizacji)

Przykład (na razie dosyć naiwny):

- zaszyfrować:
każdą literę alfabetu zastąpić numerem telefonu kogoś, czyje nazwisko zaczyna się na tą literę
- odszyfrować:
w **odwrotnej** książce telefonicznej znaleźć nazwiska odpowiadające numerom telefonów

Siła:

zależna od trudności w pozyskaniu odwrotnej książki telefonicznej
bez niej odszyfrowywanie jest beznadziejnie pracochłonne.

Szyfrowanie **RSA**



Ron Rivest



Adi Shamir



Leonard Adleman

- **kryptografia asymetryczna**
- **kryptografia klucza publicznego**

Szyfrowanie RSA

Niech p i q będą różnymi liczbami pierwszymi; i $r = pq$.

Niech $e \in [2 \dots r-1]$ będzie wzgl. pierwsze z $p-1$ i z $q-1$ (czyli z $\varphi(r)$).

Niech d — odwrotność e modulo $\varphi(r)$:

$$e \cdot d \equiv_{\varphi(r)} 1$$

Klucz jawny: para liczb (r, e) — każdy może go użyć, żeby mi przysłać zaszyfrowaną wiadomość

postać jawna wiadomości: jakaś liczba $L \in [0 \dots r-1]$

postać zaszyfrowana: reszta $C = L^e \bmod r$

Klucz tajny: para liczb (r, d) — liczby d nie ujawniam

postać zaszyfrowana wiadomości: liczba C

postać odszyfrowana: $L' = C^d \bmod r$

Szyfrowanie RSA

Niech p i q będą różnymi liczbami pierwszymi; i $r = pq$.

Niech $e \in [2 \dots r-1]$ będzie wzgl. pierwsze z $p-1$ i z $q-1$ (czyli z $\varphi(r)$).

Niech d — odwrotność e modulo $\varphi(r)$:

$$e \cdot d \equiv_{\varphi(r)} 1 \tag{1}$$

Stąd ist. k t.ż. $e \cdot d + \varphi(r) \cdot k = 1$

Czy $L' = L$? Czyli czy $(L^e)^d \equiv_r L$?

Przypadek $\text{nwd}(L, r) = 1$ Z tw. Euklidesa: $L^{\varphi(r)} \equiv_r 1$, więc

$$(L^e)^d = L^{ed} \equiv_r L^{ed} \cdot L^{\varphi(r) \cdot k} \equiv_r L^{ed + \varphi(r) \cdot k} = L^1 = L$$

Przypadek $\text{nwd}(L, r) \neq 1$ L i $r = pq$ mają wspólny dzielnik; p i q pierwsze i $L < pq$, więc $L = p$ lub $L = q$. Jeśli $L = p$, to

$$\begin{aligned} L^{ed} &= p^{ed} \equiv_p 0 \\ L &= p \equiv_p 0 \end{aligned}$$

$$p^{q-1} = p^{\varphi(q)} \equiv_q 1$$

$$\begin{aligned} L^{ed} &= p^{ed} = p^{1+\varphi(r)(-k)} = p^1 \cdot p^{\varphi(r)(-k)} \\ &= p \cdot p^{(p-1)(q-1)(-k)} = p \cdot (p^{q-1})^{(p-1)(-k)} \\ &\equiv_q p \cdot 1 = p \\ L &= p \equiv_q p \end{aligned}$$

Z chiń. tw. o resztach:

L^{ed} i L dają te same reszty modulo p i q ; p i q są różne pierwsze; więc $L^{ed} \equiv_{pq} L$

Szyfrowanie RSA

Przykład:

Niech $\begin{cases} p = 23 \\ q = 43 \\ e = 5 \end{cases}$ (czyli $r = 23 \cdot 43 = 989$).

OK, bo 23 i 43 różne pierwsze, $\text{nwd}(5, 22) = 1$ i $\text{nwd}(5, 42) = 1$,

Jak wyznaczyć d ?

$$\varphi(989) = 989 \cdot (1 - \frac{1}{23}) \cdot (1 - \frac{1}{43}) = 989 \cdot \frac{22}{23} \cdot \frac{42}{43} = 924$$

$$5 \cdot d \equiv_{924} 1 \iff d = 185$$

Zaszyfrować 587:

Klucz jawny: (989, 5)

$$587^5 = (587^2)^2 \cdot 587 \equiv_{989} 397^2 \cdot 587 \equiv_{989} 358 \cdot 587 \equiv_{989} 478$$

Odszyfrować 478:

Klucz tajny: (989, 185)

$$\begin{aligned} 478^{185} &= (478^2)^{92} \cdot 478 \equiv_{989} (25^2)^{46} \cdot 478 = (625^2)^{23} \cdot 478 \\ &\equiv_{989} (959^2)^{11} \cdot 959 \cdot 478 \equiv_{989} 900^{11} \cdot 495 = (900^2)^5 \cdot 495 \cdot 900 \\ &\equiv_{989} 9^5 \cdot 450 \equiv_{989} 698 \cdot 450 \equiv_{989} 587 \end{aligned}$$

Bezpieczeństwo szyfrowania RSA

Czy łatwo jest złamać szyfr RSA?

Znamy **publiczny** klucz (r, e) .

Do odszyfrowywania potrzebujemy klucza **prywatnego** (r, d) .

Jak wyliczyć potrzebne d , znając r i e ? — rozwiązać równanie „modulo”

$$e \cdot d \equiv_{\varphi(r)} 1$$

Jak wyliczyć $\varphi(r)$? — Wiemy, że $r = p \cdot q$ dla jakichś (**nieznanych**) pierwszych p i q ; i $\varphi(pq) = (p-1)(q-1)$.

Okazuje się, że

- **wymnożenie** dwóch dużych liczb p i q daje się łatwo wykonać; ale
- **rozłożenie na czynniki** dużej liczby r wymaga astronomicznych mocy obliczeniowych; przynajmniej wg znanych algorytmów.

Bezpieczeństwu szyfrowania RSA
mogą zagrozić komputery kwantowe...

