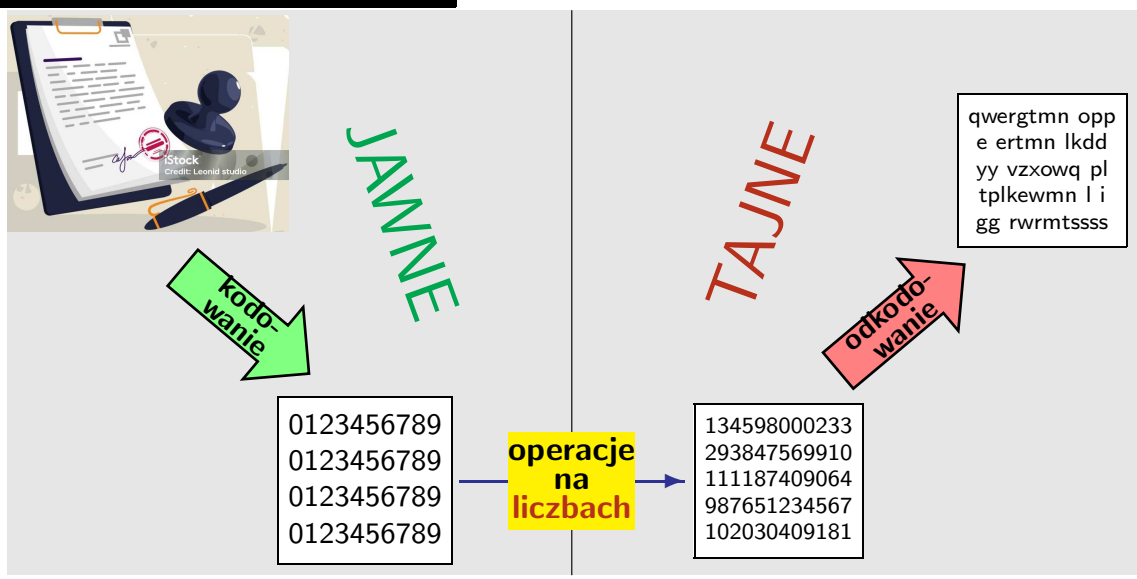
Stefan Sokołowski

KRYPTOGRAFIA

wykład 2

Inst. Informatyki Stosowanej, ANS
Elbląg, 2026/2027

Matematyka szyfrów



Celem jest, żeby

- odbiorca **wtajemniczony mógł** z tekstu tajnego odtworzyć jawny,
- odbiorca **niewtajemniczony nie mógł** z tekstu tajnego odtworzyć jawnego,
- odbiorca **wtajemniczony mógł** się upewnić, że szyfrogram pochodzi od legalnego nadawcy (wiadomość „podpisana").

Szyfrowanie z kluczem

Zasada Kerckhoffsa:

- sam algorytm szyfrowania powinien być jawny i znany wszystkim zainteresowanym,
- a bezpieczeństwo ma opierać się na nieznajomości **klucza** przez ew. napastnika.

Kiedy ktoś opuszcza grupę wtajemniczonych, to nie trzeba zmieniać algorytmu, a tylko klucz.

Podstawowe rodzaje szyfrów z kluczem:

szyfr	*symetryczny*	*asymetryczny*
	klucz K	para kluczy K_1 i K_2
szyfrowanie	$M \stackrel{E_K}{\to} C$	$M \stackrel{E_{K_1}}{\to} C$
odszyfrowywanie	$C \stackrel{D_K}{\to} M$	$C \stackrel{D_{K_2}}{\to} M$
poprawność	$D_K(E_K(M)) = M$	$D_{K_2}(E_{K_1}(M)) = M$

E — *encoding* M — *message* K — *key*
D — *decoding* C — *cryptogram*

Problem: jak bezpiecznie ustalić wspólny klucz?

Szyfrowanie blokowe

- klucz jest ciągiem bitów ustalonej długości;
- tekst **jawny:** zostaje zapisany jako jednolity **ciąg** bitów; np.

jawny:	W	I	T	A	J
ASCII:	87	73	84	65	74
binarnie:	01010111	01001001	01010100	01000001	01001010
ciąg:	0101011101001001010101000100000101001010				

- **ciąg** bitów zostaje podzielony na **bloki** równej długości, zależnej od długości klucza; tutaj — dla klucza długości 10;
- każdy blok jest szyfrowany osobno w ten sam sposób i tym samym kluczem; do odszyfrowywania używa się **tego samego klucza** (więc nadawca i odbiorca muszą się wcześniej umówić) — szyfrowanie blokowe jest *symetryczne*;
- bezpieczeństwo szyfru (trudność w złamaniu) wzrasta ze zwiększeniem długości klucza.

Przypomnienie — operacja *XOR*

$\oplus$	0	1
0	0	1
1	1	0

zwykła alternatywa „$\vee$" ma tutaj 1

XOR — *exclusive OR*, czyli dyzjunkcja, czyli alternatywa wykluczająca

$p \oplus q \;=\; 1 \iff$ dokładnie jedna z wartości p i q jest 1

$p \oplus q = q \oplus p$ — przemienność

$(p \oplus q) \oplus r = p \oplus (q \oplus r)$ — łączność

$p \oplus 0 = p$ — el. neutralny

$p \oplus p = 0$ — odwrotność

Jeśli operacją szyfrującą jest „XOR-owanie" tekstu jawnego z kluczem, to powtórne „XORowanie" z tym samym kluczem daje wyjściowy tekst jawny:

$$\begin{aligned}(\text{jawny} \oplus \text{klucz}) \oplus \text{klucz} &= \text{jawny} \oplus (\text{klucz} \oplus \text{klucz}) \\ &= \text{jawny} \oplus 0 \\ &= \text{jawny}\end{aligned}$$

Szyfrowanie blokowe przez klucz i *XOR*

Przykład: (klucz — **0010101101**)

jawny:	W	I	T	A	J
ciąg bitów:	0101011101	0010010101	0100010000	0101001010	
klucz $\times 4$:	0010101101	0010101101	0010101101	0010101101	
szyfrogram:	0111110000	0000111000	0110111101	0111100111	
bez klucza:	|	ETX	?	?	?
klucz $\times 4$:	0010101101	0010101101	0010101101	0010101101	
ciąg bitów:	0101011101	0010010101	0100010000	0101001010	
odszyfrowany:	W	I	T	A	J

Czy takie szyfrowanie jest bezpieczne?

- **Bez znajomości klucza** nie da się odszyfrować.
- Atak **brutalną siłą**: sprawdzić wszystkie możliwe klucze.
  Kluczy o długości n jest 2^n:
  $2^{10} = 1024 \qquad 2^{20} = 1\,048\,576 \qquad 2^{56} \approx 72\,000\,000\,000\,000\,000$
  rok $\approx$ 31 500 000 sek.

Szyfrowanie blokowe przez klucz i *XOR*

A co, jeśli użyty klucz jest błędny? { np. gdy zamiast **0010101101**, użyjemy **0010101100**

jawny:	W	I	T	A	J
szyfrogram:	0111110000	0000111000	0110111101	0111100111	
zły klucz $\times 4$**:**	0010101100	0010101100	0010101100	0010101100	
ciąg bitów:	0101011100	0010010100	1001000100	0101001011	
odszyfrowany:	W	[illegible]	D	E	K

Zmiana nawet jednego bitu w kluczu może uniemożliwić odszyfrowanie. Jednak *współbieżność* może umożliwić przejrzenie w realistycznym czasie wszystkich 2^n kluczy; więc umożliwić skuteczny atak brutalną siłą.

Jak się bronić?

Wielokrotne szyfrowanie różnymi kluczami? — to nic nie daje:

łączność

$$(\ldots((\text{jawny} \oplus \text{klucz}_1) \oplus \text{klucz}_2) \ldots \oplus \text{klucz}_{k-1}) \oplus \text{klucz}_k$$
$$= \text{jawny} \oplus (\text{klucz}_1 \oplus \text{klucz}_2 \ldots \oplus \text{klucz}_{k-1} \oplus \text{klucz}_k)$$

— równoważne szyfrowaniu jednym wynikowym kluczem.

Szyfrowanie zaszyfrowanego

- Zaszyfrowanie przez $\oplus$ z jakimś kluczem, a następnie z innym kluczem, nie poprawia bezpieczeństwa szyfru.
- Ale jeśli między tymi $\oplus$-owaniami zastosujemy jakieś przestawienia, to sytuacja może ulec zmianie.
- **Sieć Feistela** — algorytm szyfrowania wielorundowego z przestawieniami i „fukcjami rundy" zależnymi od kluczy:
  — Niech $k \in \mathbb{N}_{\geqslant 1}$;
  — niech $f : \{0,1\}^k \to \{0,1\}^k$ — funkcja (zależna od tajnego klucza) przekształcająca ciągi bitów długości k w ciągi bitów długości k;
  — tekst dzielimy na bloki $2k$-bitowe (długości parzystej!) i na każdym bloku wykonujemy rundę:

Szyfrowanie: | L lewa połowa bloku | R prawa połowa bloku | $\xrightarrow{f}$ | R | $L \oplus f(R)$ |

Odszyfrowywanie: | L | R | $\xrightarrow{f}$ | $R \oplus f(L)$ | L |

Dla skutecznego zaszyfrowania będą potrzebne następne rundy (z innymi f-ami), bo na razie połowa bloku nie jest zaszyfrowana.

Feistel — dwie rundy

szyfrowanie:	odszyfrowywanie
$[\,L\,|\,R\,] \xrightarrow{f} [\,R\,|\,L \oplus f(R)\,]$	$[\,L\,|\,R\,] \xrightarrow{f} [\,R \oplus f(L)\,|\,L\,]$

Przykład:

Niech $\left\{\begin{array}{ll} k = 4 & \\ f_0(c_0c_1c_2c_3) = (1010) & \text{dla dow. } c_0c_1c_2c_3 \\ f_1(c_0c_1c_2c_3) = (\bar{c}_0\bar{c}_1\bar{c}_2\bar{c}_3) & \text{— zaprzeczenia} \end{array}\right\}$

Szyfrowanie: najpierw $\xrightarrow{f_0}$, potem $\xrightarrow{f_1}$:

$$[1001|1101] \xrightarrow{f_0} [1101|1001 \oplus 1010] = [1101|0011]$$
$$\xrightarrow{f_1} [0011|1101 \oplus 1100] = [0011|0001]$$

Odszyfrowanie: najpierw $\xrightarrow{f_1}$, potem $\xrightarrow{f_0}$:

$$[0011|0001] \xrightarrow{f_1} [0001 \oplus 1100|0011] = [1101|0011]$$
$$\xrightarrow{f_0} [0011 \oplus 1010|1101] = [1001|1101]$$

to samo

Feistel — czy bezpieczny?

Jak już policzyliśmy, $[10011101] \xrightarrow{f_0} \ldots \xrightarrow{f_1} [00110001]$

ale: $[10011101] \oplus [10101100] = [00110001]$

czyli tyle samo...

Czy przejście $\xrightarrow{f_0} \ldots \xrightarrow{f_1}$ jest zawsze równoważne $\oplus$-owaniu z $[10101100]$?

Nie, bo np. $[11111111] \xrightarrow{f_0} \ldots \xrightarrow{f_1} [01010101]$ a

$$[11111111] \oplus [10101100] = [01010011]$$

Nie ma **pojedynczego** ciągu bitów, który $\oplus$-owany z **dowolnym** ciągiem dawałby to samo, co kilka rund algorytmu Feistela.

W tym sensie wielorundowe szyfrowanie Feistela jest bezpieczne.

Problem:
każda runda potrzebuje osobnego klucza; ile więc informacji strony muszą wymienić dla ustalenia sposobu komunikacji?

DES — *Data Encryption Standard*

- DES był oficjalnym standardem szyfrowania w USA od 1974 do 2005. Od 1997 wiadomo, że daje się złamać (przez kilka do kilkudziesięciu dni nieprzerwanej pracy wyspecjalizowanego komputera); dlatego 8 lat później przestał być standardem.
- DES jest blokowym szyfrem *symetrycznym* — ze znajomości klucza do szyfrowania wynika również sposób odszyfrowywania.
- DES działa na blokach 64-bitowych (8-bajtów); przekształca każdy w 64-bitowy blok szyfrogramu.
- DES działa z **jednym** kluczem 56-bitowym (7 bajtów) oraz z ustalonym sposobem generowania z niego funkcji dla 16 rund Feistela.
- DES jest opisany w bardzo wielu miejscach, np. w `https://pl.wikipedia.org/wiki/Data_Encryption_Standard`