FAKULTA INFORMATIKY A STATISTIKY

Karel, Šafr, Ph.D.

karel.safr@vse.cz

Jak prolomit šifru?

Lámání šifer pomocí statistických metod

Úvod do základů prolamování historických šifer pomocí statistických metod

V 2.1

Vysoká škola ekonomická v Praze

Obsah

- Historie šifrování a prolamování šifer
- Četnosti (n-gramy)
- Césarova šifra
- Vinegerova šifra
- Substituční šifra
  - Standardní M-H
  - M-H algoritmus Bayes
  - Gibbs sampling (varianta M-H)
  - Deterministickou greedy optimalizaci (koordinátní ascent)
  - E-M algoritmus
  - Genetický algoritmus
  - PSO algoritmus
  - Nested sampling
  - Simulované žíhání
- Prolamování moderních šifer
- PRESENT

Plán

- 1 část: 9:00 – 12:00
- 2 část: 13:00 – 15:00

FAKULTA INFORMATIKY A STATISTIKY

Karel Šafr, Ph.D.
karel.safr@vse.cz

Historie šifrování

Vysoká škola ekonomická v Praze

Historie šifrování

- Za nejstarší dochované známky šifrování z 400 let před Kristem (systém hranaté trubky a papíru ☺)
- Caesarova šifra - posun o 3 písmena v abecedě
- Al-Khalil (717-786) – Book of Cryptographic messages
  - Nejvýznamnější dílo jak v šifrování, tak v Kryptanalýze až do 2WW!!!
  - Frekvenční analýza, techniky prolomení různých šifer aj.
    - Ibn Aldan (1187-1268) o významu velikosti vzorku pro frekvenční analýzu
- O něco později se objevují šifrovací disky - cca od roku 1400
- Leon Battista Alberrti – 1467- otec západní kryptografie, polyalfabetické šifry, substituce s více permutacema

Historie šifrování

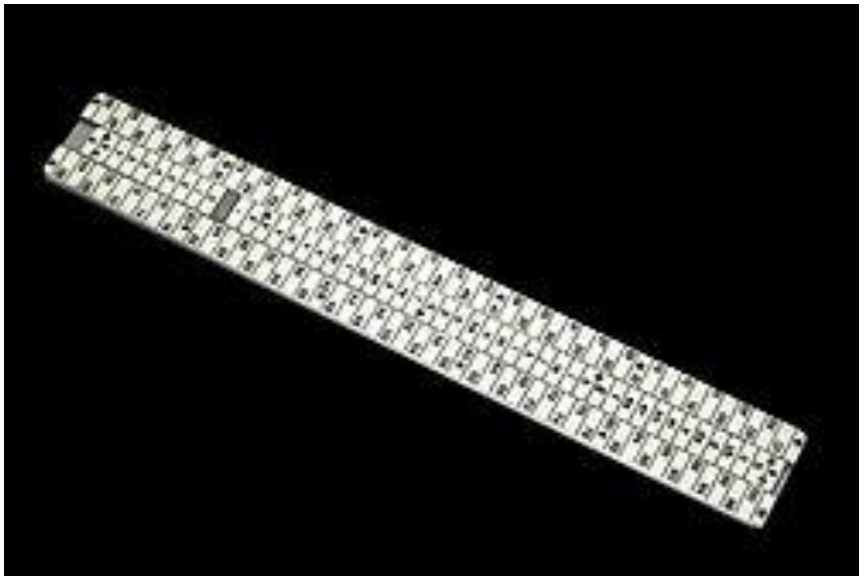
- https://en.wikipedia.org/wiki/Alberti_cipher

Historie šifrování

- Blaise de Vigenere (1523-1586) a Vigenerova šifra!!!
  - Používaná ještě **v průběhu druhé světové války!**
- V průběhu druhé světové války se začaly významně používat šifrovací přístroje na bázi rotační disků (Enigma, SIGABA, Lorenz…)
- Jejich životnost končí s příchodem moderní kryptografie, která je reakcí na narůstající výkonnost moderních počítačů – DES standard (1970)
  - Bloková šifra, operace nad bloky – permutace/transpozice na bitové úrovni
  - 2012 Brutal force attack – DES cracking machine - 768 gigakeys/sec., jakékoliv heslo do 26 hodin
  - Diferenciální kryptoanalýza (1980), Lineární kryptoanalýza (s 51% na úspěch)

Historie šifrování

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



https://en.wikipedia.org/wiki/Vigen%C3%A8re_cipher

Statistická kryptoanalýza

- Věda zabývající se metodami získání obsahu šifrovaných informací:
  - Bez znalosti informací, které jsou za běžných okolností nutné (zejména klíč)
  - Využívá statistických metod
  - Počítačová lingvistika
  - Statistické strojové učení (zejména pro moderní šifry)

- Máme i jiné, “ne zcela“ statistické přístupy používané pro moderní šifry:
  - Differential cryptanalysis
  - Linear cryptanalysis
  - Integral cryptanalysis…

Základní typy šifer

- Proudové šifry
  - Šifruje se znak po znaku tak, jak znaky přicházejí
  - Obvykle méně náročné na výpočetní zdroje
  - Náchylné na kryptanalýzu
  - Většina klasických šifer (Cesar, Vigener, Substituční šifra…)
- Blokové šifry
  - Šifruje se „blok“ dat dohromady – například 8 písmen
  - Obvykle více náročnější na výpočetní zdroje
  - Méně náchylné na kryptanalýzu (a zejména statistickou)
  - Moderní šifry – DES, AES, Blowfish, PRESENT
  - „Obvyklou“ metodou prolomení je brutal force attack

Proudové šifry | Stream ciphers - klasické

- Monoalfabetické šifry
  - Každé písmeno nezašifrovaného textu je nahrazeno jiným písmenem (a->b) buď podle nějakého pravidla, či náhodně - příkladem bude Caesarova šifra, affine šifra, bývá kombinováno s transpozicí, permutací abecedy aj..
- Polyaflabetické šifry
  - Každé písmeno nezašifrovaného textu může být zašifrováno dvěma či i více různými písmeny. Někdy se využívá kombinace více abeced (dvě a více).
- Polygrafické šifry
- Vermanova šifra (neprolomitelná?!)

Ostatní typy šifer

- Asymetrické šifry
  - Veřejný a soukromý klíč
    - Jeden umožnuje zprávu zašifrovat a druhý dešifrovat
  - RSA, PGP atd.
  - Používají se zejména v digitálním podpisu a podobných úlohách
- Hashovací funkce
  - Jednosměrné funkce používané většinou pro ověření integrity původních dat
  - Výhodou je, že délka výsledného hashe je fixní (a obvykle kratší/menší než původní objekt) -> i zde mohou paradoxně nastat problémy (MD5 atd.)
- Kvantová kryptografie, hybridní šifry / mixy šifer….

VŠE / FAKULTA INFORMATIKY A STATISTIKY

Karel Šafr, Ph.D.
karel.safr@vse.cz

Pomocné funkce a data

Vysoká škola ekonomická v Praze

Abecedy

- Existuje mnoho variant abeced. Dnes budeme používat klasickou abecedu kapitálek se znakem podtržítko „_“: značící mezeru.
  - Nic vám ale nebrání použít klasickou celou českou abecedu. Popřípadě i jiné abecedy. Pro kryptoanalýzu není předem nutné znát danou abecedu, ale je nutné ji následně identifikovat.
  - Krom samotného počtu unikátních znaků pomáhá i to, že speciální znaky se obvykle vkládají na konec abecedy, to pak může v některých šifrách způsobit, že část znaků je dešifrována správně a část špatně.
  - $\mathbb{Z}$,
  - $e_k: t \rightarrow e_k(t)$
  - $d_k: c \rightarrow d_k(c)$
  - $d_k(e_k(t)) = t$

A	B	C	D	E	F	G	H	I
0	1	2	3	4	5	6	7	8
J	K	L	M	N	O	P	Q	R
9	10	11	12	13	14	15	16	17
S	T	U	V	W	X	Y	Z	_
18	19	20	21	22	23	24	25	26

Pomocný text a abeceda

Pomocný text najdete v souboru „jlondon.txt“, jedná se o následující středně dlouhý text:

TEMNY_JEDLOVY_LES_CHMURIL_SE_PO_OBOU_STRANACH_ZAMRZLE_REKY_NEDAVNYM_VETREM_BYLY_STROMY_ZBAVENY_BILEHO_OBALU_JINOVATKY_A_ZDALO_SE_JAKO_BY_SE_OPIRALY_VZAJEMNE_O_SEBE_CERNE_A_ZLOVESTNE_V_HASNOUCIM_SVETLE_OHROMNE_TICHO_VLADLO_NAD_ZEMI_ZEME_SAMA_BYLA_PUSTINA_BEZ_ZIVOTA_BEZ_POHYBU_TAK_OSAMELA_A_CHLADNA_ZE_NALADA_JEJI_NEBYLA_ANI_NALADOU_SMUTKU_BYL_V_NI_NADECH_SMICHU_ALE_SMICHU_HROZNEJSIHO_NEZ_JAKYKOLI_SMUTEK_SMICHU_KTERY_BYL_BEZRADOSTNY_JAKO_USMEV_SFINGY_SMICHU_STUDENEHO_JAKO_MRAZ_A_TAJICIHO_V_SOBE_CHMURNOST_NEOMILNOSTI_BYLA_TO_VELITELSKA_NESDELITELNA_MOUDROST_VECNOSTI_VYSMIVAJICI_SE_MARNOSTI_ZIVOTA_A_USILI_ZIVOTA_BYLA_TO_DIVOCINA_NEZKROTNA_SEVERSKA_DIVOCINA_SE_ZMRZLYM_SRDCEM_ALE_ZIVOT_ZDE_BYL_UPROSTRED_TETO_ZEME_A_VYZYVAVY_DOLU_PO_ZAMRZLE_RECE_PLAHOCILA_SE_SMECKA_VLCICH_PSU_JEJICH_JEZATA_SRST_BYLA_POKRYTA_JINOVATKOU_DECH_MRZL_VE_VZDUCHU_JAKMILE_OPUSTIL_JEJICH_TLAMY_TRYSKAJE_V_KOTOUCICH_PARY_A_USAZOVAL_SE_NA_CHLUPECH_TEL_V_KRISTALECH_JINOVATKY_KOZENY_POSTROJ_BYL_NA_PSECH_A_PRUHY_KUZE_POJILY_JE_K_SANIM_JENZ_SE_VLEKLY_VZADU_SANE_BYLY_BEZ_OBVYKLE_SPONDI_CASTI_BYLY_ZROBENY_ZE_SILNE_BREZOVE_KURY_A_SPOCIVALY_CELOU_PLOCHOU_NA_SNEHU__PREDNI_KONEC_SANI_BYL_SPIRALOVITE_VZHURU_ZAHNUT_ABY_STLACOVAL_VSTVU_KYPREHO_SNEHU_KTERY_SE_ZVEDAL_PRED_NIM_JAKO_VLNA_NA_SANICH_BYLA_DLOUHA_A_UZKA_BEDNA_BEZPECNE_PRIVAZANA_BYLY_JESTE_JINE_VECI_NA_SANICH_PRIKRYVKY_SEKERA_HRNEC_NA_KAVU_A_PANVICKA_ALE_NEJNAPADNEJSI_BYLA_DLOUHA_A_UZKA_BEDNA_ZAUJIMAJICI_NEJVICE_MISTA

Pomocný text a abeceda

- Budeme používat následující jednu abecedu obsahující 27 znaků kódovaných do čísel jako znaky od 0 do 26. !
  - Pro šifru PRESENT ale budeme moci použít jakýkoliv UTF-8 znak.
- Speciální znak „_“, který substituuje mezeru, bude uveden jako 27 znak.
  - Pozor si na tohle musíme dát u merge a aggregate operací, které jej převádějí jinak na 1 znak před písmena
- Je zcela na vás, jakou používáte abecedu, ale ta by měla odpovídat zašifrovanému textu.
- Zároveň je nutné pro konkrétní abecedu a konkrétní jazyk si sestavit tabulku četností !

	letters	numeric
1	A	0
2	B	1
3	C	2
4	D	3
5	E	4
6	F	5
7	G	6
8	H	7
9	I	8
10	J	9
11	K	10
12	L	11
13	M	12
14	N	13
15	O	14
16	P	15
17	Q	16
18	R	17
19	S	18
20	T	19
21	U	20
22	V	21
23	W	22
24	X	23
25	Y	24
26	Z	25
27	_	26

Četnosti n-gramů

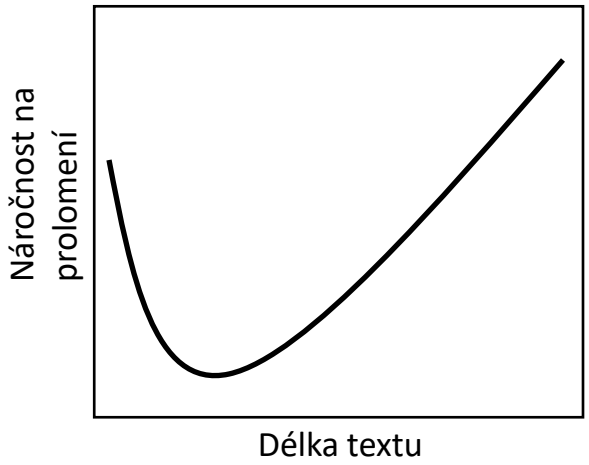
- Základním statistickým nástrojem budou pro nás běžné četnosti a relativní četnosti
- Je nutné získat četnosti, které odpovídají populaci textu, ze kterého pochází naše šifra
- V praxi jsou všechny nástroje statistického dešifrování neúčinné pro situace, kdy neznáme četnosti původních písmen.
- Všechny statistické techniky jsou postavené na pravděpodobnosti výskytu daného písmena, který chápeme jako kategoriální veličinu popřípadě na souslednosti těchto znaků.

	letters	n	f
1	_	76832	1.768754e-01
2	A	35408	8.151294e-02
3	B	5702	1.312660e-02
4	C	11913	2.742498e-02
5	D	12332	2.838956e-02
6	E	38444	8.850214e-02
7	F	454	1.045156e-03
8	G	365	8.402684e-04
9	H	8992	2.070053e-02
10	I	22309	5.135767e-02
11	J	8196	1.886805e-02
12	K	14466	3.330225e-02
13	L	19485	4.485652e-02
14	M	11927	2.745721e-02
15	N	21295	4.902333e-02
16	O	29675	6.831497e-02
17	P	13033	3.000334e-02
18	Q	5	1.151053e-05
19	R	16439	3.784431e-02
20	S	19519	4.493479e-02

	letters	n	f
1	_A	5273	1.213903e-02
2	_B	2419	5.568805e-03
3	_C	2863	6.590943e-03
4	_D	3600	8.287598e-03
5	_E	140	3.222955e-04
6	_F	135	3.107849e-04
7	_G	98	2.256068e-04
8	_H	2114	4.866662e-03
9	_I	315	7.251648e-04
10	_J	5137	1.182594e-02
11	_K	3469	7.986022e-03
12	_L	884	2.035066e-03
13	_M	3138	7.224023e-03
14	_N	7095	1.633347e-02
15	_O	2551	5.872684e-03
16	_P	8649	1.991095e-02
17	_Q	2	4.604221e-06
18	_R	2306	5.308667e-03
19	_S	8371	1.927097e-02
20	_T	5960	1.372058e-02

Náročnost výpočtů

- R není vhodný nástroj pro velmi náročné výpočty
  - Lze ale využít například spolu s tensorflow + grafickou kartou
- Vztah mezi náročností na prolomení a délkou textu má U – tvar
- Tento vztah je způsoben:
  - Při krátké délce:
    - Je dešifrování textu jednoduché
    - Málo dat, ale na výpovědní schopnost pro
    - Statistickou kryptanalýzu
  - Při dlouhé délce:
    - Dešifrování textu je náročnější na výpočet
    - Dostatečné pro výpovědní schopnost



Před samotnou analýzou je vhodné zjistit:

Složitost kryptanalýzy umocňuje neznalost:

- Použité abecedy
- Speciální znaky a interpunkce
- Metoda šifrování
- Jazyk
- Technické problémy: encoding, verze implementace šifry….

Obvykle lze ale velkou část tohoto uhodnout. V praxi jsou tyto věci často známé (veřejné standardy aj., softwary samy vkládají identifikaci dané šifry do zašifrovaného textu aj).

VŠE / FAKULTA INFORMATIKY A STATISTIKY

Karel Šafr, Ph.D.
karel.safr@vse.cz

Caesarova šifra

Vysoká škola ekonomická v Praze

Caesarova šifra

- Substituční monoalfabetická šifra
- Rodina šifer založená fixním na posunu abecedy podle předem zadané délky kroku/posunu.
- Pro dešifrování je nutné znát délku posunu.
- César ji popsal v zápiskách o válce galské, kdy využíval posunu o tři písmena.
- Tedy:
- A->D, B->E, C->F

Caesarova šifra

- Obvykle se posun značí písmenem (klíč) – kde písmeno se pak interpretuje jako číslo (n)
- Tedy pokud klíč bude b, tak znaky budou posunuté o jeden - znak **a** bude šifrován jako znak **b**
- nechť $N(x)$ bude funkce která převede charakter x na číslo a $N^{-1}(x)$ číslo naopak převede na text.
- Pak lze matematicky zapsat šifrování a dešifrování jako:
  - $e_k(t) = N^{-1}(\,(N(t) + n)\ mod\ |\mathbb{Z}|)$
  - $d_k(c) = N^{-1}(\,(N(c) - n)\ mod |\mathbb{Z}|)$

	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	_
1	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	_
2	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	_	a
3	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	_	a	b
4	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	_	a	b	c

Ukázka zašifrování:

"TEMNY_JEDLOVY_LES_CHMURIL_SE_PO_OBOU_STRANACH_ZAMRZLE_RE
KY_NEDAVNYM_VETREM_BYLY_STROMY_ZBAVENY_BILEHO_OBALU_JINOV
ATKY_A_ZDALO_SE_JAKO_BY_SE_OPIRALY_VZAJEMNE_O_SEBE_CERNE_
A_ZLOVESTNE_V_HASNOUCIM_SVETLE_OHROMNE_TICHO_VLADLO_NAD_Z
EMI_ZEME_SAMA_BYLA_PUSTINA_BEZ_ZIVOTA_BEZ_POHYBU_TAK_OSAM
ELA_A_CHLADNA_ZE_NALADA_JEJI_NEBYLA_ANI_NALADOU_SMUTKU "

"VGOP_BLGFNQX_BNGUBEJOWTKNBUGBRQBQDQWBUVTCPCEJBACOTANGBTG
M_BPGFCXP_OBXGVTGOBD_N_BUVTQO_BADCXGP_BDKNGJQBQDCNWBLKPQX
CVM_BCBAFCNQBUGBLCMQBD_BUGBQRKTCN_BXACLGOPGBQBUGDGBEGTPGB
CBANQXGUVPGBXBJCUPQWEKOBUXGVNGBQJTQOPGBVKEJQBXNCFNQBPCFBA
GOKBAGOGBUCOCBD_NCBRWUVKPCBDGABAKXQVCBDGABRQJ_DWBVCMBQUCO
GNCBCBEJNCFPCBAGBPCNCFCBLGLKBPGD_NCBCPKBPCNCFQWBUOWVMW "

Implementace caesarovy šifry:

```
caesar_encrypt  ← function(text, key = num_to_char(n), n =
char_to_num(key)){
  numvec_to_char((char_to_numvec(text) + n) %% nrow(alphabet))
}

caesar_decrypt  ← function(text, key = num_to_char(n), n =
char_to_num(key)){
  numvec_to_char((char_to_numvec(text) - n) %% nrow(alphabet))
}
```

Možnosti prolomení

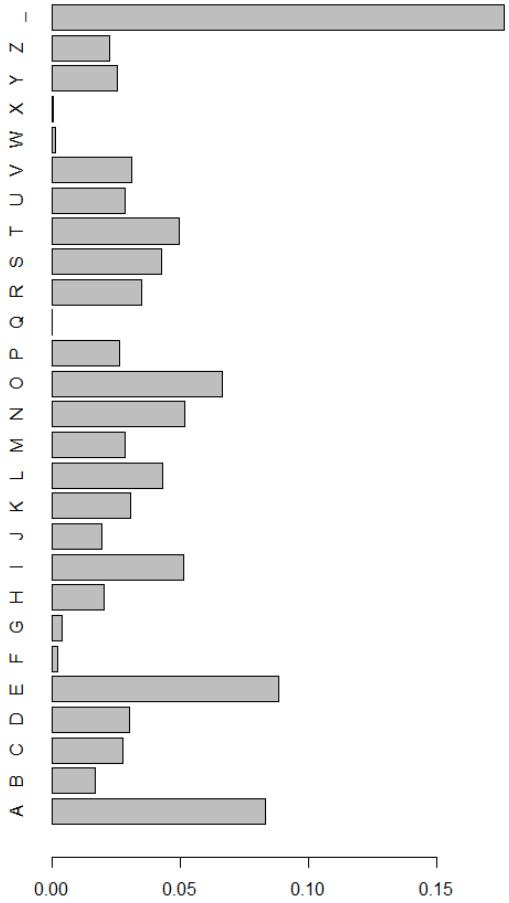
Jednoduchá na prolomení:

a) Lze ji ručně prolomit pomocí prostého zkoušení nejčetnějších písmen v zašifrovaném textu na nejvíce četné písmeno v abecedě pro identifikaci posunu.
b) Existuje i algoritmus na nejkratší vzdálenost mezi písmeny
    a) Čistě numerický postup vycházející z měření vzdálenosti mezi vybranými písmeny, lze použít na jakékoliv vzdálenosti, ale nejčastěji se používá na ty nejvíce četnější
c) Lze použít frekvenční analýzu

Ruční prolomení

"VGOP_BLGFNQX_BNGUBEJOWTKNBUGBR
QBQDQWBUVTCPCEJBACOTANGBTGM_BPG
FCXP_OBXGVTGOBD_N_BUVTQO_BADCXG
P_BDKNGJQBQDCNWBLKPQXCVM_BCBAFC
NQBUGBLCMQBD_BUGBQRKTCN_BXACLGO
PGBQBUGDGBEGTPGBCBANQXGUVPGBXBJ
CUPQWEKOBUXGVNGBQJTQOPGBVKEJQBX
NCFNQBPCFBAGOKBAGOGBUCOCBD_NCBR
WUVKPCBDGABAKXQVCBDGABRQJ_DWBVC
MBQUCOGNCBCBEJNCFPCBAGBPCNCFCBL
GLKBPGD_NCBCPKBPCNCFQWBUOWVMW„

- Nejčetnější v textu je písmeno B a podle teoretických četností (text Čapek) je to mezera, pak písmeno „e“ a písmeno „a“
- Délka posunu bude nejspíše jedním z těchto rozdílů



Frekvenční analýza

- Četnosti písmen různých variant dešifrovaného textu vs. Čapek.
- Optické porovnání není ideální, a tak se obvykle používá chi-kvadrát test dobré shody.

! Pozor, text obvykle nesplňuje předpoklady testování hypotéz v chi-kvadrát testu dobré shody ($n_i > 5$ | 3, $N > 30$).

Chí-kvadrát test

Vyjdeme ze standardního chí-kvadrát testu dobré shody:

Chí-kvadrát test dobré shody

H_0 a H_1	Testové kritérium	Kritický obor
H_0: $\pi_j = \pi_{0,j}$ $j = 1, \ldots, k$ H_1: non H_0	$G = \sum_{j=1}^{k} \frac{(n_j - n\pi_{0,j})^2}{n\pi_{0,j}}$ $G \approx \chi^2(k-1)$	$W_\alpha = \{g; g \geq \chi^2_{1-\alpha}\}$ $n\pi_j \geq 5$

Myšlenka je pak stejná pro našich 27 písmen abecedy (0 pozice!):

$$G = \sum_{i=0}^{26} \frac{(C_i - E_i)^2}{E_i}$$

Kde C_i je počet i-tého písmena v zašifrovaném textu, E_i je očekávaný počet písmena dle teorie – text reprezentující teoretickou četnost písmen populace ze které pochází zašifrovaná zpráva.

Chi-kvadrát test

- Řešením je takové písmeno, které má nejnižší chi-kvadrát hodnotu
- V praxi se ale často testuje několik nejnižších hodnot
- Na tohle má vliv délka zašifrovaného textu, abeceda, technické a jiné netradiční výrazy v původním nezašifrovaném textu.

	letters	numeric	chi
1	A	0	7.258606e+11
2	B	1	2.731141e+11
3	C	2	7.068812e+11
4	D	3	8.445367e+11
5	E	4	1.471481e+11
6	F	5	1.374281e+08
7	G	6	3.020624e+11
8	H	7	7.182927e+11
9	I	8	9.012492e+11
10	J	9	9.484164e+11
11	K	10	5.218689e+11
12	L	11	4.428775e+10
13	M	12	4.284438e+10
14	N	13	5.722030e+11
15	O	14	4.271728e+11
16	P	15	7.132430e+12
17	Q	16	1.988653e+12
18	R	17	2.316303e+11
19	S	18	3.506710e+11
20	T	19	1.553267e+11
21	U	20	1.942955e+12
22	V	21	6.732974e+10
23	W	22	6.620965e+10
24	X	23	3.225546e+11
25	Y	24	7.462356e+11
26	Z	25	2.458047e+11
27	_	26	3.157940e+11

Sloupcová transpozice:

- Způsob jak zesílit šifrování
- Zvolíme **klíčové slovo**, zapíšeme text do bloku pod toto klíčové slovo
- Podle abecedního pořadí si doplníme čísla sloupců a následně čteme zprávu po sloupcích!

klíč:	j	a	n	e	k
poř.ab.	3	1	5	2	4
->	b	y	l	p	o
	z	d	n	i	v
	e	c	e	r	p
	r	v	n	i	m
	a	j	q	a	w

- bylpozdnivecerprvnimaj
- Se zašifruje jako:
- ydcvjpiriabzeraovpmwlnenq
- Text byl doplněn do bloku o náhodná písmena (qaw)
- Jednoduché na prolomení, ale používá se pro ztížení ostatních šifer,
- Princip transpozice sloupců (míchání dat) se používá i v moderních šifrách (např. AES, PRESENT…)
- Častá situace, kdy frekvenční analýza (ngram = 1) vrací výsledek, který není lidsky čitelný, ale očividně správný. Bigramy nejsou v souladu s ngram =1 !
  - Řešením je použít frekvenční analýzu s ngram =1 a následně hledat transpozici, která sedí na bigramy!

Sloupcová transpozice:

- Počet možných kombinací sloupců pak odpovídá délce klíče:

$$Key_n!$$

- Klíč o délce 8 pak vede na nutnost otestovat 40320 kombinací a klíč o 9 počtu písmen na 362 880.
- Z tohoto důvodu byla historicky sloupcová transpozice velmi oblíbená, zejména v kombinaci s Vigenerovou šifrou či substituční šifrou (viz navazující slide)
- Sloupcovou tranpospozici ale ze snadno prolomit pomocí metody kombinující Markovovi řetězce a Monte-Carlo metodu!
  - Budeme si toto ukazovat pro substituční šifru. Pro sloupcovou transpozici je to jen analogie.

VŠE / FAKULTA INFORMATIKY A STATISTIKY

Karel Šafr, Ph.D.
karel.safr@vse.cz

Vigenèrova šifra

Vysoká škola ekonomická v Praze

Vigenèrova šifra

- Vigenerova šifra se liší od Caesarovy šifry v tom, že délka klíče nemusí být jeden znak, ale je obvykle právě větší než 1 znak.
- Pro první znak textu, který se má zašifrovat, se použije první znak klíče, pro druhý druhý znak klíče… ve chvíli, kdy dojde k využití všech znaků klíče, tak je klíč znova recyklován. Tedy klíč se cyklicky opakuje.
- Pokud by byla délka klíče stejná jako délka textu a klíč byl náhodný (a byl jen jednorázově použit), tak šifra je neprolomitelná (opravdu ☺).
- Vigenerova šifra s klíčem o shodné délce textu je vlastně Vernamova šifra (1917) – one- pad šifra
  - Náhodný klíč, neopakovaný (jednorázový) klíč, stejně dlouhý jako zpráva

One-pad šifra – Vermanova šifra

- Sověti používali one-pad šifru až do 50. let či snad déle v běžné vojenské a špionážní komunikaci.
- Problém byl, že jejich klíč nebyl náhodný !!
- Během druhé světové války USA pochopily systém textů, ze kterých pochází jejich klíč a podařilo se odhadnout frekvenci očekávaných posunů!!!
- Většina šifer tak z let 1940-1945 byla tímto způsobem prolomena!

- Pokud je správně použitá (Vernamova šifra), tak je zcela neprolomitelná.
- Vernamova šifra se používá (stále!) pro různé specializované účely, například za studené války se používala pro šifrování tzv. „horké linky“.
- Tradičně se používala/á pro spojení s agenty v poli:
  jednorázová tabulka -> pak zničit, jednoduché na použití -> tužka + papír

Vigenerova šifra - příklad šifrování

Text:	A	H	O	J	A	H	O	J
klíč:	A	B	C	A	B	C	A	B
shift:	+0	+1	+2	+0	+1	+2	+0	+1
šifra:	A	I	Q	J	B	J	O	K

Alternativní pohled:

Slovo: AHOJAHOJ

Klíč:	A	B	C	A	B	C
Sloupce:						
-> čtení:	A	H	O	0	1	2
	J	A	H	0	1	2
	O	J		0	1	

prevod:	A	B	C	
	A	I	Q	-> čtení:
	J	B	J	AIQJBJOK
	O	K		

```
vigener_encrypt<-function(text, key){
 text<-char_to_vec(text)
 v.key=char_to_vec(key)
 for(i in 1:(nchar(key))){
  s = seq(from=i, to = length(text), by = nchar(key))
  text[s] = char_to_vec(caesar_encrypt(vec_to_char(text[s]),key=v.key[i]))
 }
 return(vec_to_char(text))
}

vigener_decrypt<-function(text, key){
 text<-char_to_vec(text)
 v.key=char_to_vec(key)
 for(i in 1:(nchar(key))){
  s = seq(from=i, to = length(text), by = nchar(key))
  text[s] = char_to_vec(caesar_decrypt(vec_to_char(text[s]),key=v.key[i]))
 }
 return(vec_to_char(text))
}
```

Prolomení Vigenèrovy šifry

- Prolomení Vigenerovy šifry má dva kroky:
  - Zjištění délky klíče (m)
  - Vytřídění dat dle odpovídajících sekvencí a aplikace metod prolomení Caesarovy šifry
- Zjištění délky klíče je asi nejproblematičtější část - kdyby délka klíče odpovídala délce textu a klíč byl náhodný, tak text je neprolomitelný !
  - V takovou situaci lze najít jakoukoliv interpretaci textu jako stejně pravděpodobnou!
- Máme několik způsobů jak postupovat při hledání klíče:
  - **Frekvenční analýza s typováním délky klíče**
  - **Kasinskiho metoda**
  - **Friedmanův test – index koincidence**

Frekvenční analýza pro získání délky klíče

Zvolíme počet sloupců = m

- Rozdělíme text do m sloupců (či sekvencí)
- Následně zjistíme posun písmen s nejnižším rozdílem od teoretické četnosti písmen pro každý sloupec zvlášť.
- Vykreslíme / uložíme souhrnný chí-kvadrát test a dílčí posuny (heslo)

Můžeme postupovat pro jiný počet sloupců potom, co projdeme všechny kandidáty, je délka hesla a heslo u počtu sloupců s nejnižší hodnotou souhrnného testu.

Frekvenční analýza pro získání délky klíče

	shift	klic	chi
9	10	JACKLONDON	1,37E+08
4	5	VAJKA	3,15E+10
11	12	P_DAIOL_CDBD	4,84E+10
7	8	OJFAUXAD	6,45E+10
8	9	ABDPKDACA	7,01E+10
10	11	EODJBAXJQAQ	7,54E+10
5	6	W_DVCO	8,78E+10
6	7	DAQOGAC	9,03E+10
3	4	U_CD	1,17E+11
1	2	CD	1,35E+11
2	3	D_D	1,39E+11

Frekvenční analýza pro získání délky klíče

```
prolom_vigener<-function(enc, M_seq=2:15){
  keys<-data.frame(matrix(0,nrow=0,ncol=3,dimnames=list(c(),c("shift","klic","chi"))))
  text = char_to_numvec(enc)
  n = length(text)
  for(shift in M_seq){
    sloupce<-list()
    for(i in 1:shift){
      sloupce[[i]]<-text[seq(from = i, to =n, by = shift)]
    }
    keystream=lapply(sloupce,function(x) prolom_caesar(numvec_to_char(x)))
    # aplly keystream and decode vigener and count cetnosti,
    key=paste0(unlist(keystream),collapse="")
    g.text= vigener_decrypt(enc, key)

    cet = cetnosti(g.text)
    cet = merge(CZfreq,cet, by = "letters",suffixes = c(".teor",".emp"),all=TRUE)
    cet[is.na(cet)] = 0

    chi=sum(((cet$n.emp-cet$f.teor*sum(cet$n.emp))**2)/cet$f.teor*sum(cet$n.emp))
    keys=rbind(keys,data.frame(shift=shift, klic=key, chi =chi))

  }

  keys[order(keys$chi),]
}
```

Kasiskiho metoda - Friedrich W. Kasiski

- Kasiski vychází z toho, že v textu se s určitou pravděpodobností **vyskytne jedno slovo** (či n-gram) **víckrát** a je **vysoká šance, že bude zašifrován stejnou částí klíče** (může jít jen o část slova, či více - jedna část pokrývající části dvou slov)
- Pak zjistíme počet znaků mezi těmito n-gramy (od prvního znaku prvního n-gramu po poslední znak před druhým n-gramem) a **počet sloupců** (délka klíče) **je jedním z celočíselných dělitelů tohoto čísla**.
  - Toto je logický závěr, jinak by nemohlo opakováním klíče dojít k zašifrování textu na stejné znaky.
  - Soustředíme se na dlouhé řetězce stejných znaků (7 a více), u kterých je menší pravděpodobnost, že se jedná o dva rozdílné texty, které byly ale zašifrované stejně.

Kasiskiho metoda

```
table_ngram←function(x,n){
  x=table(ngram_word(x,n))
  x[x>1]
}
divisors ← function(x){
  y ← seq_len(x)
  y[ x%%y == 0 ]
}

tn=table(ngram_word(enc,9))

tn[tn>1]

kasiski←function(repword,text){
  r=gregexpr(repword,text)[[1]][1:2]
  rozdil = r[2]-r[1]

  divisors(rozdil)
}

kasiski("IJKXZINWY",enc)
kasiski("CMVDOVKKP",enc)
```

Friedmanův test - index koincidence

- Friedmanův test je po Kasiského metodě další způsob, jak zjistit délku klíče
- Obvykle se tyto dvě metody kombinují!
- Friedmanův test nám vrací délku klíče, která je **nejvíce pravděpodobná**!
  - Oproti tomu **Kasiského test nám vrací všechny možné délky klíče.**
- Friedmanův test vychází z indexu koincidence –
  - Pravděpodobnost, že vybereme konkrétní písmeno z náhodného textu, je **1/27.**
  - ALE u přirozeného jazyka neodpovídá index koincidence náhodnému textu ☺.
  - Pak můžeme spočítat index koincidence pro různé abecedy.

Friedmanův index koincidence

- Index koincidence nám umožňuje odhadnout pravděpodobnost jevu, že dvě náhodně vybraná písmena z textu budou stejná. Pokud n_k znamená četnost k-tého písmene v textu pak můžeme pro daný text(T) napsat následující vzoreček:

$$IC(T) = \sum_{k=0}^{N-1} \frac{n_k(n_k - 1)}{n(n-1)}$$

Což lze pro velký počet písmen zjednodušit na přibližné řešení:

$$IC(T) \cong \sum_{k=0}^{N-1} p_k^2$$

Kde p_k značí relativní četnost k-tého písmene z daného textu.

Friedmanův index koincidence

- Pro zcela náhodný text, kde každé písmeno má četnost 1/27 (pro naši abecedu) pak platí, že index koincidence je 0.037037.
  - A tedy platí, že pokud jsme z náhodného textu vybrali náhodně písmeno a že v dalším tahu jej vytáhneme znovu a bude 1/27*0,037037, což je, jako bychom napsali: (1/27)*(1/27)
- Tento fakt ale neplatí pro přirozené jazyky a musíme si pro daný jazyk a typ textu vypočíst vlastní index koincidence

Friedmanův index koincidence

- Rozepišme si, co víme:
  - Pravděpodobnost toho, že dvě písmena jsou v rozdílných sloupcích je (m- délka klíče):
    - $\mathbf{1 - 1/m}$
  - Pravděpodobnost toho, že dvě písmena v různých sloupcích jsou identická pro zcela náhodný text je pak:
    - 1/27 = 0.037
  - Pravděpodobnost toho, že dvě náhodně vybraná písmena v různých sloupcích šifry jsou identické je pak:
    - **(1-1/m)*1/27**
  - Takže:

$$IC^C \approx \frac{1}{m} IC^L + \left(1 - \frac{1}{m}\right)\frac{1}{27}$$

Lze odhad délky klíče získat jako:

$$m \approx \frac{IC^L - \frac{1}{27}}{IC^c - \frac{1}{27}}$$

Kde IC^C je index koincidence pro danou šifru a IC^L pro daný jazyk, Metoda ale funguje přibližně!!!

Friedmanův test - index koincidence

```
# Friedman:
ICcz=sum(CZfreq$f**2)
# odhad délky (+-):
friedman←function(text, IC=ICcz,n=27){
  m= (IC-1/n) / (sum(cetnosti(text)$f**2)-1/n)
  m
}

friedman(text=enc)
7.41
# → tedy 5, nebo 10
```

Zvažujeme tedy jen tyto klíče:

shift	klic	chi
2	10 JACKLONDON	137428104
1	5 VAJKA	31486748243

VŠE / FAKULTA INFORMATIKY A STATISTIKY

Karel Šafr, Ph.D.
karel.safr@vse.cz

Permutace – substituční šifra

Vysoká škola ekonomická v Praze

Monoalfabetická substituční šifra

- Někdy též tzv. permutační šifra
- Za každé písmeno je podle předem neznámého klíče zvoleno jiné písmeno
- Dochází k zachování frekvence četnosti písmen, proto se v minulosti kombinovalo s transpozicí či/a Vigenerovou šifrou
  - S ohledem na výpočetní náročnost zůstávala dlouho v „provozu“. Šifra byla nahrazována v praktickém použití (diplomacie / vojenství) až pod tlakem blížící se druhé světové války, ale někde se používala až do 50. let
  - O slabosti této šifry se vědělo ale již od raných dob (viz. první arabští autoři)
    - Tajná abeceda + Transpozice | klíč zaručovali dlouhou dobu neprolomitelnost
  - Pokud se permutace abecedy použila jen jednou a zpráva nebyla dost dlouhá (tisíce znaků), pak ji bylo problém i tak prolomit – mnoho znaků má podobnou četnost, viz. Ibn Aldan (1187-1268)

Substituční šifra

- Blokové šifrovací nástroje (Enigma, SIGBA…) jsou v podstatě substituční šifrou kombinovanou s posunem písmen v abecedě po každém písemně. Tedy provádím substituční šifru na substituční šifře…a tak dále
- Při blokové šifře dochází ale k rotaci každého bloku podle předem daného klíče, díky tomu je jedno písmeno pokaždé zašifrované jinak (než se otočí úplně celé válce) – tedy to, jak je signál (písmeno) z jednoho bloku zašifrované v druhém, se mění.
- Počet bloků ovlivňuje počet možností toho, jak je jedno písmeno zašifrované.

→ Každé písmeno je šifrované jinou permutací abecedy podle předem definovaného pohybu rotorů.

Prolomení substituční šifry

- Máme vícero způsobů, jak prolomit substituční šifru:
  - Brutal force - vede ale na permutaci 27 písmen abecedy
  - Standardní frekvenční analýza má problém - hodně písmen je obdobně četných. Aby byla frekvenční analýza efektivní, potřebovali bychom texty o délce ~5-10 tisíce a více písmen.
    - I v středně dlouhých textech 500-1500 znaků jsou četnosti méně četných písmen často podobné ve většině abeced.
    - I tak by to vedlo na velký počet pokusů, a navíc by bylo nutné ručně procházet hodně textu.
    - Dá se vylepšit slovníkovou metodou!

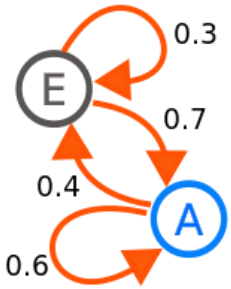
    -> Ukážeme si

Prolomení substituční šifry

- Jedno z výpočetně nejefektivnějších postupů prolomení šifry vychází z kombinaci více statistických metod.
- Vyjděme z následujícího postupu:
  - Prvně vygenerujme náhodný klíč (k_0) a ověříme jak daný text je přijatelný / věrohodný (p_0)
  - Následně procházíme smyčkou:
    - Vygenerujme aktualizovaný klíč k_i- a to tak že pro hodíme náhodně dva prvky klíče (dvě písmena) mezi sebou
    - vypočteme pro tento klíč věrohodnost textu (p_i)
    - Následně:
      - Pokud $p_i > p_{i-1}$: Pokračuj
      - Jinak: $p_i = p_{i-1}$; $k_i = k_{i-1}$
- Jedná se v podstatě o Monte-Carlo algoritmus

Prolomení substituční šifry

- Problémem je, jak vyhodnotit přijatelnost (věrohodnost) klíče.
- Pokud bychom použili standardní frekvenční analýzu, tak získáme nesmyslný text - metoda nebude konvergovat a obvykle nepřekročí hranici 20-30% úspěšného zatřídění písmen.
- Zde přichází na řadu přechodová matice z Markovových řetězců:
  - Markovův řetězec popisuje diskrétní stochastický proces, pro který platí, že pravděpodobnost přechodu ze současného do následujícího stavu závisí pouze na současném stavu.
  - Obvykle se vizualizuje přechodovou maticí
    - pravděpodobností, či naměřenou absolutní
  - **Tedy změříme si, jak často po jednom konkrétním písmenu v českém textu následuje jiné konkrétní písmeno, a konfrontujeme to s četností v našem dešifrovaném textu**



0.3	0.7
0.4	0.6

Měření věrohodnosti textu - plausability

- Věrohodnostní funkce:

$$PL(k) = \prod \left(M_{ij}\right)^{\widehat{M_{ij}}}$$

- Kde M_{ij} jsou prvky přechodové matice v absolutním vyjádření pro naměřené hodnoty populace a $\widehat{M_{ij}}$ jsou naměřené hodnoty přechodu pro daný text
- Pak nejlepší typ klíče by měl být největší věrohodnost
- Funkce v této formě je nevypočitatelná – musí se logaritmovat
- Je nutné přičíst 1 ke každému nulovému prvku přechodové absolutní matice vypočtené z teoretického vzorového textu!!! – kvůli logaritmu

Prolomení substituční šifry

- Často se stává, že tato metoda se zastaví v lokálním extrému.
- Problémů se zaseknutím v lokálním extrému se lze vyvarovat:
  - Například podmínkou, že v případě, kdy nedojde k výměně klíče, dojde s určitou pravděpodobností k výměně i tak při dostatečně vysoké pravděpodobnosti (např. podíl věrohodností musí být větší než 0.9 a zároveň k tomu dojde s pravděpodobností 1 %)
  - Nebo použitím genetického algoritmu:
    - Po určitém počtu iterací dojde k přechodu do nové generace
    - Obvykle část jedinců přejde rovnou, část se zahodí a další noví jedinci vzniknou jako potomci stávajících (mutací, kombinací se stávajícími aj.)

```
[1] "4920. iterace, log plaus:10328; spravne: 25"
[1] "4930. iterace, log plaus:10328; spravne: 25"
[1] "4940. iterace, log plaus:10328; spravne: 25"
[1] "4950. iterace, log plaus:10328; spravne: 25"
[1] "4960. iterace, log plaus:10328; spravne: 25"
[1] "4970. iterace, log plaus:10328; spravne: 25"
[1] "4980. iterace, log plaus:10328; spravne: 25"
[1] "4990. iterace, log plaus:10328; spravne: 25"
[1] "5000. iterace, log plaus:10328; spravne: 25"
[1] "------------------------------------------------------------"
> tv=table(res$guess_c,key)
> sum(diag(tv))/sum(tv)
[1] 0.9259259
> res$gtext_c
[1] "TEMNY_JEDLOVY_LES_CHMURIL_SE_PO_OBOU_STRANACH_ZAMRZLE_REKY_NEDAVNYM_VETREM_BYLY_STROMY_ZBAVENY_BILEHO_OBALU_JINOVATKY_A_ZDALO_SE_JAKO_BY_SE_OPIRALY_VZAJEMNE_O_SEBE_CERNE_A_ZLOVESTNE_V_HASNOUCIM_SVETLE_OHROMNE_TICHO_VLADLO_NAD_ZEMI_ZEME_SAMA_BYLA_PUSTINA_BEZ_ZIVOTA_BEZ_POHYBU_TAK_OSAMELA_A_CHLADNA_ZE_NALADA_JEJI_NEBYLA_ANI_NALADOU_SMUTKU_BYL_V_NI_NADECH_SMICHU_ALE_SMICHU_HROZNEJSIHO_NEZ_JAKYKOLI_SMUTEK_SMICHU_KTERY_BYL_BEZRADOSTNY_JAKO_USMEV_SFINGY_SMICHU_STUDENEHO_JAKO_MRAZ_A_TAJICIHO_V_SOBE_CHMURNOST_NEOMILNOSTI_BYLA_TO_VELITELSKA_NESDELITELNA_MOUDROST_VECNOSTI_VYSMIVAJICI_SE_MARNOSTI_ZIVOTA_A_USILI_ZIVOTA_BYLA_TO_DIVOCINA_NEZKROTNA_SEVERSKA_DIVOCINA_SE_ZMRZLYM_SRDCEM_ALE_ZIVOT_ZDE_BYL_UPROSTRED_TETO_ZEME_A_VYZYVAVY_DOLU_PO_ZAMRZLE_RECE_PLAHOCILA_SE_SMECKA_VLCICH_PSU_JEJICH_JEZATA_SRST_BYLA_POKRYTA_JINOVATKOU_DECH_MRZL_VE_VZDUCHU_JAKMILE_OPUSTIL_JEJICH_TLAMY_TRYSKAJE_V_KOTOUCICH_PARY_A_USAZOVAL_SE_NA_CHLUPECH_TEL_V_KRISTALECH_JINOVATKY_KOZENY_POSTROJ_BYL_NA_PSECH_A_PRUHY_KUZE_POJILY_JE_K_SANIM_JENZ_SE_VLEKLY_VZADU_SANE_BYLY_BEZ_OBVYKLE_SPONDI_CASTI_BYLY_ZROBENY_ZE_SILNE_BREZOVE_KURY_A_SPOCIVALY_CELOU_PLOCHOU_NA_SNEHU__PREDNI_KONEC_SANI_BYL_SPIRALOVITE_VZHURU_ZAHNUT_ABY_STLACOVAL_VSTVU_KYPREHO_SNEHU_KTERY_SE_ZVEDAL_PRED_NIM_JAKO_VLNA_NA_SANICH_BYLA_DLOUHA_A_UZKA_BEDNA_BEZPECNE_PRIVAZANA_BYLY_JESTE_JINE_VECI_NA_SANICH_PRIKRYVKY_SEKERA_HRNEC_NA_KAVU_A_PANVICKA_ALE_NEJNAPADNEJSI_BYLA_DLOUHA_A_UZKA_BEDNA_ZAUJIMAJICI_NEJVICE_MISTA"
> 
```

Možné další přístupy?

- Většina ostatních přístupů využívá nějakou metodu permutace klíče a statistických / optimalizačních metod
- Základem je ale obvykle měření věrohodnosti

Možné další přístupy?

Algoritmus	Oblast matematiky/statistiky	Klíčové vlastnosti
Metropolis-Hastings (MH)	Markov Chain Monte Carlo, Bayes	Stochastický, flexibilní, posteriorní odhady
Gibbs Sampling	Markov Chain Monte Carlo, Bayes	Jednodušší speciální případ MH algoritmu
Koordinátní ascent (variační inference)	Deterministická optimalizace (greedy přístup)	Rychlá konvergence, náchylná na lokální optima
EM algoritmus	Bayesovská inference, pravděpodobnostní metody	Iterativní, střídá výpočet posterioru a optimalizaci
Genetický algoritmus	Evoluční výpočty, evoluční biologie	Robustní, paralelizovatelný, explorativní
Particle Swarm Optimization (PSO)	Heuristická optimalizace, kolektivní chování	Snadná implementace, rychlá komunikace částic
Nested Sampling	Bayesovská inference, statistická fyzika	Adaptivní explorace posteriorního prostoru
Simulované žíhání (Simulated Annealing)	Stochastická optimalizace, statistická fyzika	Schopnost uniknout lokálním optimům, snadná implementace

M-H algoritmus a Bayes

Bayesovské metody umožňují snadno zakomponovat naše předpoklady (priory) o textu, jako například:

- Pravděpodobnost jednotlivých písmen, dvojic nebo trojic v cílovém jazyce (například četnost „E“, „ST“, „TH“ v angličtině, nebo „O“, „E“, „A“ v češtině).
- Informace o klíčích (například rovnoměrné či nerovnoměrné apriorní pravděpodobnosti substitucí).
- Díky tomu lze dobře odhadnout jak původní text, tak i klíč.

$$P(klíč\ |šifrovaný\ text\ ) \propto = P(šifrovaný\ text\ |klíč)\ * P(klíč)$$

PRIOR: P(klíč) - Naše počáteční přesvědčení o pravděpodobnosti konkrétních klíčů (typicky rovnoměrné rozdělení, když nemáme žádnou další informaci).

LIKELIHOOD: P(šifrovaný text | klíč) - Pravděpodobnost, že konkrétní text vznikne z určitého klíče (využíváme jazykový model).

POSTERIOR: P(klíč | šiforvaný text) - Naše aktualizované přesvědčení o pravděpodobnosti klíče poté, co jsme viděli šifrovaný text.

M-H algoritmus a Bayes

Dále v M-H algoritmu budeme využívat Bayesovský akceptační poměr:

$$A = \frac{P(nový\ text) * P(nový\ klíč)}{P(starý\ text) * P(starý\ klíč)} * \frac{q(starý\ klíč\ |nový\ klíč)}{q(nový\ klíč|starý\ klíč)}$$

Pro symetrické návrhové rozdělení (swap dvou hodnot v klíči):

$$A = \frac{P(nový\ text) * P(nový\ klíč)}{P(starý\ text) * P(starý\ klíč)}$$

P(textu) obecně měříme věrohodností – bigramy /trigramy.

Když nemáme informaci o klíči P(klíč) – tak se volí jednoduše – bez informací nemůžeme zvýhodňovat jeden klíč před jiným. Pak

P(klíč)= konstanta , pro všechny klíče

$$A = \frac{P(nový\ text)}{P(starý\ text)}$$

M-H algoritmus a Bayes

- **Bayesovská inference** – odhad posteriorního rozdělení.
- **Markov Chain Monte Carlo (MCMC)** – skupina pravděpodobnostních metod založených na generování Markovských řetězců.

Základní myšlenka:

- Metropolis–Hastings (MH) algoritmus slouží k simulaci vzorků z cílového pravděpodobnostního rozdělení, které neumíme přímo vzorkovat nebo je velmi komplikované.
- MH algoritmus generuje řetězec vzorků (například klíčů k šifře), jehož distribuce postupně konverguje k hledanému rozdělení (zde distribuci pravděpodobnosti klíčů dle jejich plausibility).

Klíčové vlastnosti:

- MH algoritmus je obecný, nevyžaduje žádnou speciální formu distribuce.
- Umožňuje i přijetí méně plausibilních řešení, čímž se lépe vyhýbá lokálním optimům.
- Funguje dobře i v rozsáhlých prostorech řešení (vysoká dimenzionalita klíče u šifer).

M-H algoritmus a Bayes

- **Inicializace**:
  - Začneme s náhodně zvoleným klíčem (počáteční řešení).
- **Návrh nového řešení** (návrhový krok):
  - V každé iteraci vytvoříme kandidátní řešení tak, že:
  - Náhodně vybereme dva prvky v aktuálním klíči a tyto dva prvky navzájem prohodíme.
  - Tato malá změna vede ke vzniku nového kandidátního klíče.
- **Výpočet plausibility (věrohodnosti)**:
  - Pro obě řešení (aktuální i kandidátní) vypočteme plausibilitu podle pravděpodobnostního modelu (např. bigramového modelu).
  - Plausibilita odpovídá pravděpodobnosti, že daný klíč produkuje smysluplný dešifrovaný text.
- **Rozhodnutí o přijetí či zamítnutí nového řešení**:
  - Kandidátní řešení přijmeme podle tzv. Metropolisova kritéria. Pravděpodobnost přijetí kandidáta je určena vztahem:
    - **A = min(1; věrohodnost nového / věrohodnost starého klíče)**
  - Pokud nový klíč má vyšší plausibilitu, je akceptován vždy. Pokud má nižší plausibilitu, je stále možné jej akceptovat s určitou malou pravděpodobností. To umožňuje algoritmu uniknout z lokálních optim.
- **Opakování procesu:** Kroky opakujeme po dostatečný počet iterací, dokud algoritmus nekonverguje k stabilnímu řešení (klíči).

Gibbs sampling

Bayesovská inference – metoda pro aproximaci posteriorního rozdělení.

Markov Chain Monte Carlo (MCMC) – pravděpodobnostní techniky založené na Markovských řetězcích.

Co je Gibbs sampling:

- Speciální varianta Metropolis–Hastings algoritmu, kde vzorkujeme každou proměnnou podmíněně vzhledem ke všem ostatním proměnným.
- Cílem je generovat vzorky ze složitých vícerozměrných distribucí, které neumíme snadno vzorkovat přímo.
- Gibbs sampling využívá strukturu modelu, čímž výrazně usnadňuje výpočet posteriorních pravděpodobností.

Specifika použití u prolomení šifer:

- Řešením (klíčem) je permutace (uspořádání substitucí písmen).
- Algoritmus vzorkuje jednotlivé prvky klíče (substituce) postupně a podmíněně, což umožňuje efektivnější pohyb v prostoru řešení.
- Vhodný pro situace, kdy podmíněné pravděpodobnosti lze spočítat relativně rychle.

Gibbs sampling

- **Inicializace:**
  - Nastavíme náhodně počáteční klíč (počáteční substituci).
- **Iterační krok Gibbsova sampleru:**
  - Postupně projdeme všechna písmena v klíči (1 až 27, pokud používáme abecedu s 26 znaky + mezera).
- **Pro každé písmeno v klíči zvlášť:**
  - Fixujeme všechny substituce kromě aktuálně vzorkované.
  - Pro vzorkovanou substituci postupně otestujeme všech 27 možných substitucí tím, že každou potenciální substituci dosadíme do aktuální pozice.
  - Pro každou takovou substituci vypočteme její plausibilitu (věrohodnost), tj. pravděpodobnost vygenerovaného textu podle bigramového jazykového modelu.
- **Výpočet posteriorních pravděpodobností:**
  - Získáme vektor plausibility pro všechny substituce.
  - Normalizujeme jej do pravděpodobností (za použití logaritmických hodnot kvůli numerické stabilitě):
    - $P_j = \frac{\exp(\log plausability_j - \max \log plausability_j)}{\sum_k \exp(logplausability_k - \max \log plausability_k)}$
    - Vzorkování nové substituce:
      - Náhodně vybereme substituci podle těchto vypočtených pravděpodobností.Nastavíme tuto vybranou substituci do aktuální pozice v klíči.
- **Opakování postupu:**Opakujeme proces přes všechny substituce v klíči, čímž dokončíme jednu iteraci Gibbs sampleru. -> Proces opakujeme po dostatečný počet iterací, dokud algoritmus nekonverguje k relativně stabilnímu klíči.

Gibbs sampling

Výhody Gibbsova vzorkování:

- Jednodušší implementace podmíněného vzorkování než obecný MH algoritmus.
- Efektivní při využití struktury problému (např. podmíněné pravděpodobnosti lze snadno spočítat).
- Dobrá schopnost procházet prostorem řešení díky postupnému a podmíněnému vzorkování.

Nevýhody Gibbsova vzorkování:

- Může být pomalejší, pokud výpočet podmíněných pravděpodobností je náročný.
- Vyžaduje dobré pochopení struktury modelu a jeho podmíněných distribucí.

Praktické poznámky k implementaci:

- Pro numerickou stabilitu vždy používejte logaritmy pravděpodobností.
- Počet iterací nastavujte dostatečně velký pro dosažení konvergence.
- Sledujte průběžné hodnoty plausibility pro kontrolu konvergence.

Deterministické algoritmy – greedy optimalizace

Oblast matematiky:

- **Optimalizace** – hledání maxima či minima cílové funkce.
- **Deterministické optimalizační metody** – narozdíl od stochastických metod jsou tyto přístupy přesně definované a opakovatelné.

Co je greedy optimalizace a koordinátní ascent:

- „Greedy“ metoda („hladová“) v každém kroku algoritmu vybírá lokálně nejlepší možný krok s cílem maximalizace cílové funkce (v našem případě plausibility).
- **Koordinátní ascent** je speciální případ greedy optimalizace, kdy optimalizujeme řešení po jedné proměnné (souřadnici) v každém kroku a ostatní držíme konstantní.
- Metoda postupně zlepšuje řešení deterministicky – vždy vybírá nejlepší dostupnou změnu a tuto změnu provede.

Specifika u substituční šifry:

- Klíč je zde permutace znaků (prohození znaků v abecedě).
- Koordinátní ascent optimalizuje substituční klíč postupně, vždy jednu substituci za druhou, a provádí nejlepší dostupný „swap“ (prohození dvou substitucí).
- Po každém swapu dochází k jednoznačnému zlepšení plausibility, nebo algoritmus končí, pokud lepší substituci již nelze najít (konvergence).

Deterministické algoritmy – greedy optimalizace

- **Inicializace:**
  - Náhodně zvolíme počáteční klíč (substituční permutaci).
- **Iterativní optimalizace (koordinační ascent):**
  - Opakujeme cyklus, dokud nedojde ke konvergenci (žádné další zlepšení není možné).
- **Pro každou substituční pozici v klíči:**
  - Postupně vybereme aktuální substituční pozici klíče (1 až 27).
  - Pro každou možnou jinou substituční pozici v klíči (celkem 27 možností):
    - Provedeme testovací „swap“ substitucí mezi těmito dvěma pozicemi.
    - Dešifrujeme text s nově vzniklým klíčem.
    - Spočteme plausibilitu (log-plausibilitu) tohoto testovacího klíče.
  - Vybereme takový swap, který maximalizuje plausibilitu textu.
  - Pokud existuje swap s lepší plausibilitou, provedeme jej.
  - Pokud žádný swap nevede ke zlepšení, ponecháme původní klíč a označíme, že v tomto kole nedošlo ke změně.
- **Kontrola konvergence:**
  - Pokud žádný ze všech substitučních kroků v celé jedné iteraci nevede ke zlepšení (žádná změna), algoritmus dosáhl lokálního optima a ukončí se.

EM algoritmus

Statistika / pravděpodobnostní modely (Bayesovské a ne-Bayesovské metody).
Metoda je známá jako **Expectation-Maximization (EM)** algoritmus:

- Používá se k odhadu parametrů pravděpodobnostních modelů při výskytu skrytých (latentních) proměnných nebo chybějících dat.

Základní myšlenka EM algoritmu - EM algoritmus iterativně střídá dva kroky:

- **E-krok (Expectation):**
  - Na základě současných odhadů parametrů odhadujeme (aproximujeme) rozdělení nepozorovaných („latentních“) proměnných.
  - Tím získáváme očekávané hodnoty těchto nepozorovaných proměnných či doplňujeme chybějící data.
- **M-krok (Maximization):**
  - Na základě těchto očekávaných hodnot maximalizujeme věrohodnost (pravděpodobnost pozorovaných dat), čímž získáváme nové odhady parametrů modelu.

- Postupným střídáním těchto dvou kroků se algoritmus iterativně zlepšuje, až konverguje k (lokálnímu) maximu věrohodnosti.

Aplikace EM algoritmu na substituční šifru (základní myšlenka):

- Klíč k substituční šifře lze chápat jako „parametr“, který chceme odhadnout.
- „Skryté“ proměnné jsou zde správná písmena původního textu (nevidíme je přímo).
- EM algoritmus tedy postupně odhaduje pravděpodobnosti jednotlivých substitucí, optimalizuje odhad klíče a iterativně vylepšuje dešifrovaný text.

EM algoritmus

EM algoritmus v kontextu substituční šifry:

- **E-krok (Expectation)**:
  - Pomocí aktuálního klíče odhadneme (aproximujeme) posteriorní pravděpodobnosti skutečných písmen textu.
  - V naší praktické implementaci se tyto posteriorní pravděpodobnosti aproximují deterministickým odhadem textu pomocí aktuálního klíče (hard přiřazení).
  - Tím získáváme dešifrovaný text (tedy doplníme chybějící informaci o správném textu).
- **M-krok (Maximization)**:
  - Vezmeme text z E-kroku (doplněná „latentní data“) a snažíme se najít klíč, který maximalizuje plausibilitu (věrohodnost) tohoto textu podle jazykového modelu (bigramové matice).
  - Hledáme lepší substituční klíč „greedy“ optimalizací: provádíme sérii swapů substitucí a vždy vybíráme tu, která nejvíce zvýší plausibilitu textu.
  - Výsledkem M-kroku je vylepšený klíč.
- **Iterativní opakování:**
  - Tyto dva kroky (E a M) se střídají a iterují tak dlouho, dokud nedojde ke konvergenci – tj. dokud se plausibilita dále nezvyšuje, nebo dokud nenastane předem stanovený počet iterací.

Vlastnosti EM algoritmu v této úloze:

- Algoritmus je deterministický, vede k postupnému vylepšování řešení.
- Garantuje konvergenci k lokálnímu optimu plausibility textu.
- Velmi citlivý na počáteční odhad klíče, vhodné je více náhodných restartů.

Genetické algoritmy

Evoluční algoritmy a metaheuristiky

- Inspirován biologickou evolucí (Darwinův princip: přežití nejvhodnějších jedinců).

Co je genetický algoritmus?

- Optimalizační metoda založená na přírodním výběru:
  - Kandidátní řešení („jedinci“) tvoří **populaci**.
  - Jedinci mají přiřazeno **fitness** (míra kvality řešení).
  - Lepší jedinci mají vyšší pravděpodobnost přežít a předat „geny“ (parametry řešení) do další generace.
- Používají se tři základní operace:
  - **Selekce** (výběr jedinců pro reprodukci)
  - **Křížení** (kombinace parametrů rodičů)
  - **Mutace** (náhodná změna parametrů jedince)

Kontext prolomení substituční šifry:

- Každý jedinec je klíč substituční šifry (permutace písmen).
- Fitness jedince se spočítá jako plausibilita dešifrovaného textu (pomocí bigramů).
- Algoritmus postupně vylepšuje populaci klíčů, dokud nenajde vysoce pravděpodobné řešení (pravý klíč).

Genetické algoritmy

Základní koncepty v kontextu šifer:

- **Jedinec** (chromozom) = kandidátní klíč (permutace abecedy).
- **Populace** = množina klíčů, typicky velikost 20–200 jedinců.
- **Fitness** = logaritmická pravděpodobnost textu dešifrovaného daným klíčem (plausibilita založená na bigramovém modelu).

Kroky genetického algoritmu (velmi detailně):

1. **Inicializace:**
   Generuje se počáteční náhodná populace klíčů.
   Vyhodnotí se fitness každého jedince (klíče).
2. **Selekce (výběr rodičů):**
   Jedinci jsou seřazeni podle fitness (lepší fitness = vyšší šance reprodukce).
   Typické metody selekce:
   Elitismus: nejlepší jedinci automaticky přecházejí do další generace.
   Roulette Wheel (ruleta): pravděpodobnost výběru úměrná fitness.
   Turnajová selekce: zvolíme malé skupiny jedinců a z každé skupiny vybereme nejlepšího jako rodiče.
3. **Křížení (crossover):**
   Z rodičů se vytváří nový potomek kombinací jejich permutací.
   Používá se například **jednobodový crossover**:
   Rozdělíme permutaci rodičů na dvě části.
   Potomek dostane první část od prvního rodiče, druhou část od druhého rodiče, přičemž zachováváme platnost permutace (žádná duplicita písmen).
4. **Mutace:**
   Náhodná změna permutace (swap dvou písmen).
   Pomáhá zachovat diverzitu v populaci, zabránit uvíznutí v lokálních optimech.
5. **Nová generace:**
   Opakujeme kroky selekce, křížení a mutace, dokud nedosáhneme dostatečně dobrého řešení nebo maximálního počtu generací.

PSO – Particle Swart Optimization

Metaheuristiky, optimalizační algoritmy inspirované přírodou

- Často používaná metoda v oblasti umělé inteligence a strojového učení pro optimalizaci složitých funkcí s mnoha parametry.

Co je Particle Swarm Optimization (PSO)?

- PSO je algoritmus inspirovaný chováním hejn ptáků nebo rybích škol:
  - Hejno částic se pohybuje v prostoru možných řešení (stavový prostor).
  - Každá částice reprezentuje kandidátní řešení problému.
  - Částice se pohybují tak, že sledují nejlepší pozici, kterou samy dosáhly (**personal best**), a také nejlepší pozici dosaženou celým hejnem (**global best**).

Principy PSO:

- **Částice** má svou pozici (řešení) a rychlost (způsob změny řešení).
- V každé iteraci částice aktualizují svou pozici podle pravidel, která berou v úvahu:
  - **Vlastní zkušenost** (personal best)
  - **Kolektivní znalosti hejna** (global best)
- Částice postupně konvergují směrem k optimálnímu řešení v prostoru.

Využití pro prolomení substituční šifry:

- Prostor řešení tvoří permutace klíčů substituční šifry.
- Částice reprezentují kandidátní klíče.
- Fitness (kvalita řešení) = plausibilita textu po dešifrování (bigramový model).

PSO – Particle Swart Optimization

Definice prvků PSO v kontextu šifry:

- **Částice** = kandidátní klíče k šifře (permutace písmen).
- **Pozice částice** = konkrétní permutace substitucí (aktuální klíč).
- **Rychlost částice** = série swapů písmen, které provedeme na aktuálním klíči k dosažení nového řešení.

Princip aktualizace částic:

V každé iteraci částice:

1. **Zhodnotí fitness** aktuálního klíče (dešifruje text, spočítá plausibilitu).
2. **Aktualizují personal best (pBest)**, pokud aktuální klíč má lepší fitness než nejlepší dříve nalezené řešení touto částicí.
3. **Aktualizují global best (gBest)**, pokud je nalezen lepší klíč než dosavadní nejlepší řešení celého hejna.
4. **Generují novou rychlost**:
    1. Část rychlosti je generována směrem k vlastnímu pBest (částice se snaží vrátit k nejlepšímu řešení, které zná).
    2. Část rychlosti je generována směrem ke globálnímu gBest (částice následují úspěšné řešení celého hejna).
5. **Aplikují rychlost** (provede série swapů písmen, čímž se získá nový kandidátní klíč).

Krok za krokem aktualizace částice (detailněji):

- Identifikuj písmena, která se liší mezi současným klíčem a personal best nebo global best.
- Vyber náhodně páry těchto písmen a aplikuj je jako swapy:
    - To představuje „pohyb částice“ směrem k lepším řešením.
- Pravděpodobnost a intenzita tohoto pohybu je řízena parametrem **α (alpha)**, který určuje míru, do jaké částice následují globální trend.

Nested sampling

- **Bayesovská statistika a pravděpodobnostní výpočetní metody**
- Původně navrženo Johnem Skillingem (2004) pro efektivní aproximaci marginální evidence (integrálu evidence, normovací konstanty v Bayesově větě).

Co je Nested Sampling?

- Metoda pro výpočet **Bayesovské evidence** a vzorkování z komplikovaných posteriorních rozdělení.
- Je založena na postupném vzorkování a nahrazování bodů v prostoru parametrů (řešení) tak, aby iterativně zkoumala prostor s rostoucí plausibilitou.
- Narozdíl od jiných algoritmů jako Metropolis-Hastings nebo Gibbs sampling, **Nested sampling systematicky prohledává prostor podle úrovní plausibility**, nikoliv náhodně podle posteriorní distribuce.

Kontext prolomení substituční šifry:

- Prostor řešení = všechny možné permutace substitučních klíčů (velký diskrétní prostor).
- Každý klíč má přiřazenou hodnotu plausibility (logaritmická pravděpodobnost textu dešifrovaného daným klíčem dle bigramového modelu).
- Algoritmus iterativně zlepšuje skupinu kandidátních klíčů (částic) tak, že v každém kroku odstraní nejméně vhodný klíč a nahradí ho novým, lepším klíčem. Tím postupně zvyšuje dolní hranici plausibility všech klíčů v souboru.

Nested Sampling

- **Populace (částice)** – množina kandidátních klíčů, každý má spočítanou plausibilitu.
- **Úroveň plausibility** – aktuální hranice plausibility daná nejhorším jedincem v populaci.
- **Iterativní vylepšování populace** – neustále se zvyšuje dolní mez plausibility tím, že nejhorší částici odstraňujeme a nahrazujeme novou částicí s vyšší plausibilitou.

Kroky algoritmu Nested Sampling:

1. **Inicializace:**
    1. Náhodně vygenerujeme populaci (např. 10–50 částic), každá částice je permutací substitučního klíče.
    2. Každá částice má přiřazenou hodnotu plausibility.
2. **Hlavní smyčka algoritmu:**
    1. Najdeme **nejhorší částici** (nejnižší plausibilita) v aktuální populaci.
    2. Stanovíme tuto nejhorší plausibilitu jako aktuální **hranici plausibility**.
    3. Odstraníme nejhorší částici z populace.
3. **Generování nové částice (vnořené vzorkování):**
    1. Vytvoříme novou částici tak, že náhodně vybereme existující částici z populace a provedeme drobnou modifikaci jejího klíče (např. náhodně prohodíme dva symboly v permutaci).
    2. Spočteme plausibilitu nové částice.
    3. Opakujeme generování, dokud nemáme částici s plausibilitou **vyšší než aktuální hranice** (ta nejhorší, kterou jsme odstranili).
4. **Nahrazení odstraněné částice:**
    1. Novou částici vložíme do populace místo odstraněné.
    2. Tím se postupně „zvedá“ minimální úroveň plausibility populace a algoritmus směřuje ke stále lepším řešením.
5. **Opakování:**
    1. Celý postup opakujeme předem daný počet iterací (typicky tisíce iterací).
    2. Algoritmus postupně konverguje k prostoru nejvyšší plausibility.

Simulované žíhání

- **Stochastická optimalizace** (náhodnost v rozhodovacích krocích)
- Inspirace v termodynamice a fyzice (proces chlazení a krystalizace látek)

Základní myšlenka:

- Metoda je inspirována procesem pomalého ochlazování („žíhání“) materiálu, kdy látka postupně přechází ze stavu vyšší energie do nižšího energetického stavu (stabilnější konfigurace).
- Optimalizace je prováděna pomocí řízené náhody a postupně se snižující teploty, která určuje pravděpodobnost přijetí horšího řešení.
- Na začátku (vysoká teplota) algoritmus snadno přijímá horší řešení, aby prozkoumal širší prostor možných řešení a unikl z lokálních optim.
- Postupně (s klesající teplotou) se algoritmus stává více deterministickým, upřednostňuje pouze lepší řešení a sbližuje se s lokálním/globálním optimem.

Kontext prolamování substituční šifry:

- Prostor řešení tvoří všechny možné permutace substitučního klíče (27! možností).
- Cílem je nalézt permutaci klíče s maximální **plausibilitou** (hodnota pravděpodobnosti textu na základě bigramového jazykového modelu).
- Energetický stav (který minimalizujeme či maximalizujeme) odpovídá plausibilitě dekódovaného textu.

Simulované žíhání

- **Fyzika:**
  - Původní myšlenka simulovaného žíhání pochází z metalurgie, kde se kovy zahřejí na vysokou teplotu, aby molekuly volně prozkoumaly různé konfigurace (stav s vyšší energií). -> Poté se materiál postupně ochlazuje (žíhání), což umožňuje molekulám stabilizovat se v nejnižším možném energetickém stavu (stabilní krystalická struktura).
- **V optimalizaci / matematice:**
  - V optimalizaci nahradíme pojem „energie“ cílovou (optimalizační) funkcí (např. plausibilitou v kryptografii).
  - „Vysoká teplota“ znamená vysokou ochotu přijímat i horší řešení, což umožňuje algoritmu efektivně prozkoumávat prostor možných řešení a překonávat lokální maxima.)
  - Postupné snižování teploty vede k tomu, že algoritmus se stále více zaměřuje na blízké okolí aktuálně nejlepšího řešení a nakonec konverguje do (lokálního či globálního) optima.
- **Proč to ale funguje?!**
  - Vysoká počáteční teplota → algoritmus může „přeskočit“ lokální optima.
  - Nízká koncová teplota → algoritmus se stabilizuje v nejlepším dosaženém optimu.
  - Postupné snižování pravděpodobnosti přijetí horších řešení vede k tomu, že řešení může projít i „méně pravděpodobnými“ stavy, aby nakonec dospělo k lepšímu řešení.

Simulované žíhání

Stav (klíč):
•Aktuální permutace klíče (každá permutace představuje jedno řešení).
Energie (plausibilita):
•Vyhodnocuje „kvalitu“ aktuálního řešení.
•Cílem je maximalizovat plausibilitu (minimalizovat energii, negativní plausibilitu).
Teplota (řídící parametr algoritmu):
•Začínáme na vysoké hodnotě teploty T_{init} , která umožňuje přijmout horší řešení s vyšší pravděpodobností.
•Teplota postupně klesá podle určitého „chladicího schématu“ až k nízké finální teplotě T_{final}.
Rozhodovací kritérium (Metropolisovo kritérium):
•Rozhoduje, zda přijmeme nové řešení s plausibilitou p_{new} oproti současnému řešení s plausibilitou $p_{current}$:

$$pokud\ \ p_{new} > p_{current}\ přijmu\ vždy$$

$$pokud\ \ p_{new} < p_{current}\ přijmu\ s\ pravděpodobností\ e^{\frac{p_{new}-p_{cur}}{T}}$$

Simulované žíhání

Jak algoritmus funguje krok za krokem v kontextu substituční šifry:

1. Náhodně inicializujeme substituční klíč.
2. Spočteme plausibilitu odpovídajícího dešifrovaného textu.
3. Náhodně vybereme dvě písmena v klíči a prohodíme je (nový kandidátní stav).
4. Vyhodnotíme plausibilitu tohoto nového stavu.
5. Použijeme Metropolisovo kritérium k rozhodnutí, zda přijmeme nový klíč.
6. Snižujeme teplotu podle schématu (např. exponenciálně).
7. Opakujeme kroky 3–6 až do dosažení finální teploty.

Závěrem k substituční šifře

- **Prolomení substituční šifry** pomocí různých matematických a statistických algoritmů.
- Zkoumali jsme efektivitu metod pro dešifrování založených na statistické optimalizaci a Bayesovských metodách.

Jak jsme postupovali:

1. Nejprve jsme detailně pochopili problém substituční šifry jako optimalizační úlohu.
2. Prostor možných řešení představuje všechny permutace klíče (celkem $27! \approx 10^{28}$).
3. Použili jsme **bigramový jazykový model** jako míru věrohodnosti (plausibility) dešifrovaného textu.
4. Implementovali jsme a detailně analyzovali různé algoritmy pro prohledávání prostoru řešení.

S čím jsme pracovali:

- **Zašifrovaný text**
- **Bigramové pravděpodobnosti (jazykový model)**
- **Bayesovský přístup** – posteriorní pravděpodobnosti klíče
- **Stochastické optimalizační metody**

Závěrem k substituční šifře

- **Statistická inference a optimalizace** jsou efektivní nástroje pro prolomení substitučních šifer.
- **Bayesovské metody (MH, Gibbs, EM)** umožňují přímočarý a přirozený přístup, využívající pravděpodobnostní strukturu dat.
- **Heuristické metody (GA, PSO, Simulované žíhání)** nabízejí rychlé a efektivní algoritmy, které zvládají komplikované prostory řešení a umožňují uniknout lokálním optimům.

Závěrem k substituční šifře

Metoda	Oblast statistiky/matematiky	Typ aktualizace	Posteriorní aproximace
Metropolis-Hastings	Bayesovská inference, MCMC	Stochastická	Jeden řetězec vzorků
Gibbsův sampler	Bayesovská inference, MCMC	Stochastická	Podmíněné vzorkování
Greedy (variační)	Optimalizace, variační inference	Deterministická	Bodová (mode)
EM algoritmus	Statistická inference, ML	Deterministická	Bodová (mode)
Genetický algoritmus	Evoluční optimalizace	Stochastická	Populační vzorky
Particle Swarm Opt.	Heuristická optimalizace	Stochastická	Populační vzorky
Nested sampling	Bayesovská inference, evidence	Stochastická	Celý posterior
Simulované žíhání	Stochastická optimalizace, statistická fyzika	Stochastická	Jeden řetězec vzorků (bodová aproximace maxima)

VŠE / FAKULTA INFORMATIKY A STATISTIKY

Karel Šafr, Ph.D.
karel.safr@vse.cz

Moderní šifry – bloková šifra

Vysoká škola ekonomická v Praze

Blokové šifry

- Typ šifer, kde se zpráva k zašifrování zpracovává po částech – v blokách
- Obvykle se používají pro data v bitovém vyjádření a počet bitů označuje délku bloku: 16 bit, 32 bit, 64 bit... 265 bit
  - Říká nám kolik 0/1 bude zašifrováno „pospolu“.
- Blokové šifry vycházejí z principů klasických šifer a využívají všechno, co jsme doposud viděli:
  - Aplikace klíče (používá se pro bit XOR operace – exkluzivní disjunkce)
  - Míchání dat (forma transpozice)
  - Aplikace substitučních funkcí po částech bitů (vlastně permutace – substituční šifra)

1 XOR 1 = 0
1 XOR 0 = 1
0 XOR 1 = 1
0 XOR 0 = 0

Délka bloku n, délka klíče k

N-bit dlouhé bloky hodnot, např. 8 bit: [0, 1, 0, 0, 0, 1, 1,0]

2^n možných vstupů, tedy pro 8-bit dlouhý blok máme 256 možných vstupů dat (UTF -8 ☺)

- Malé n – může způsobit, že na vstupu bude stejný text a objeví se frekvence
- Velké n – náročné na výpočet

Některé algoritmy řeší toto dilema tak, že přidají náhodnou hodnotu do každého bloku.

- Malé k – hrozí brutal force attack
- Velké k – obvykle vede na větší náročnost zašifrování
- ???Uživatelem zadaná délka hesla??? -> expanze hesla do požadované délky ☹

Prolamování moderních šifer

V moderním šifrování se využívají zejména blokové šifry, na jejich prolomení byly vymyšleny dva základní přístupy a jedná se o:

- Lineární kryptanalýza (Mitsuru Matsui - FEAL šifra 1992 + DES)
- Diferenciální kryptanalýza (Eli Biham a Adi Shamir – 80s – DES)

- Tyto dva přístupy lze použít pro jak pro blokové šifry, tak i pro streamy šifer.
- Většina přístupů nevede na přímé prolomení šifry, ale na snížení náročnosti provedení brutal force útoku, což platí i pro statistickou kryptanalýzu.

Co dělá moderní šifry bezpečnými - SPN

Co by měla umět ideálně šifra?

- Diffusion: každý bit vstupu a klíče musí mít vliv na každý bit šifrované zprávy
- Confusion: vztah mezi každým bitem vstupu a výstupu musí být komplikovaný.

Dosáhnout toho můžeme pomocí SPN

- Substituce – používáme lookup tabulky (nelineární, náhodné aj.) a malé.
- Permutace – mícháme pořadí bitů v bloku tak, aby každý bit vstoupil do každé lookup tabulky a byl ovlivněn všemi bity klíče.

-> Toto se aplikuje po blokách v opakování. Dále je důležité poznamenat, že se v iteracích mění i klíč!

Permutace/substituce/ rotace klíče – round key

PRESENT – 1 blok

0101 0101 1001 0000 1010 1011 0011 1001 0101 0101 1001 1010 0000 1110 1001 0010

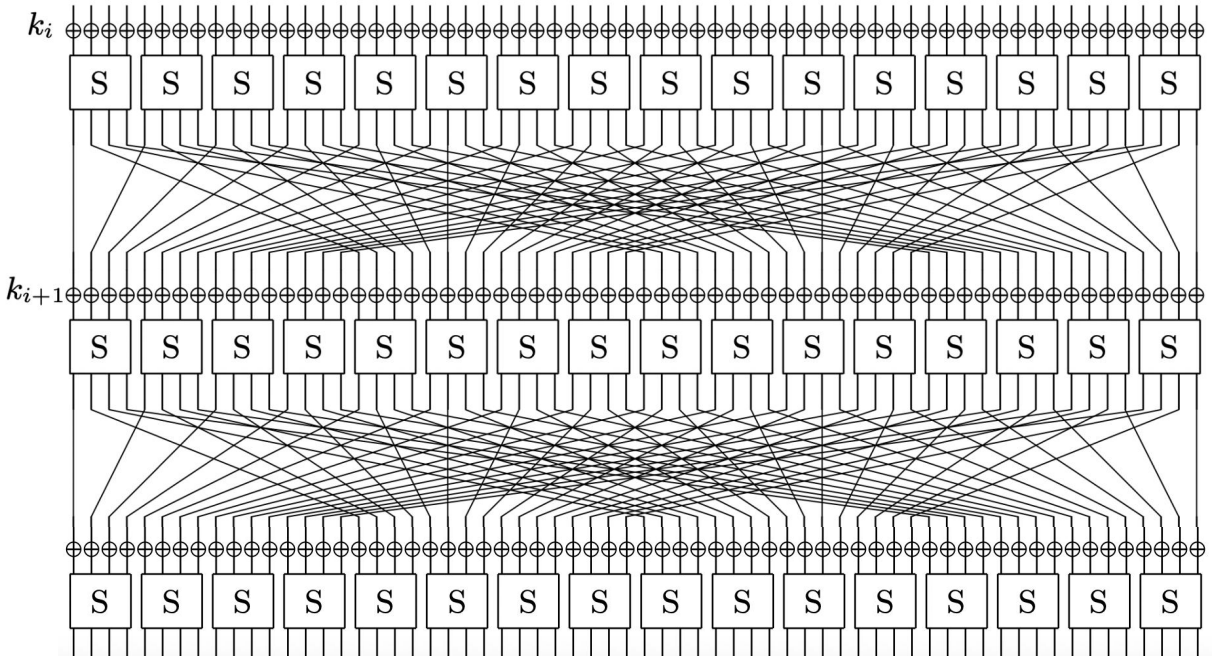
- Zpracovává 64-bit dlouhé bloky.
- Kombinuje vrstvy:
  - aplikace klíče (xor)
  - substituce 4bitových bloků zprávy (16 možných kombinací v substitučních/lookup tabulkách) – tedy 16 substitučních tabulek.
  - Míchání bitů v bloku tak, že po 2. kole každý bit z první S tabulky vstoupil do každé jiné S tabulky a ovlivnil tak všechny ostatní bity.
  - Rundovní klíč – rotace a změna klíče po vrstvách. Využívá 128 bit dlouhý klíč, který se posouvá při každém kole o 61 bitů doleva, 8 bitů prochází dvěma S tabulkami. A část bitů vpravo se násobí číslem dané iterace (po 5 bitech) – rundovní číslo. Z celého klíče se využívá jen 64 znaků vlevo .

  Má 32 kol - rund

1 XOR 1 = 0
1 XOR 0 = 1
0 XOR 1 = 1
0 XOR 0 = 0

PRESENT – design zpracování bloku

Zpráva **0111** 0010 1000 0010 1100 1010 1010 0010 0010 1010 1000 0010 0101 0010 1010 0010
Round key 0010 1010 1000 0010 0101 0010 0111 0010 1001 1010 1101 0010 1111 0010 0010 0010
zpráva 0101 1000 0000 0000 1001 1000 1101 0000 1011 0000 0101 0000 1010 0000 1000 0000



N A S E T A J E
78 65 83 69 84 65 74 69

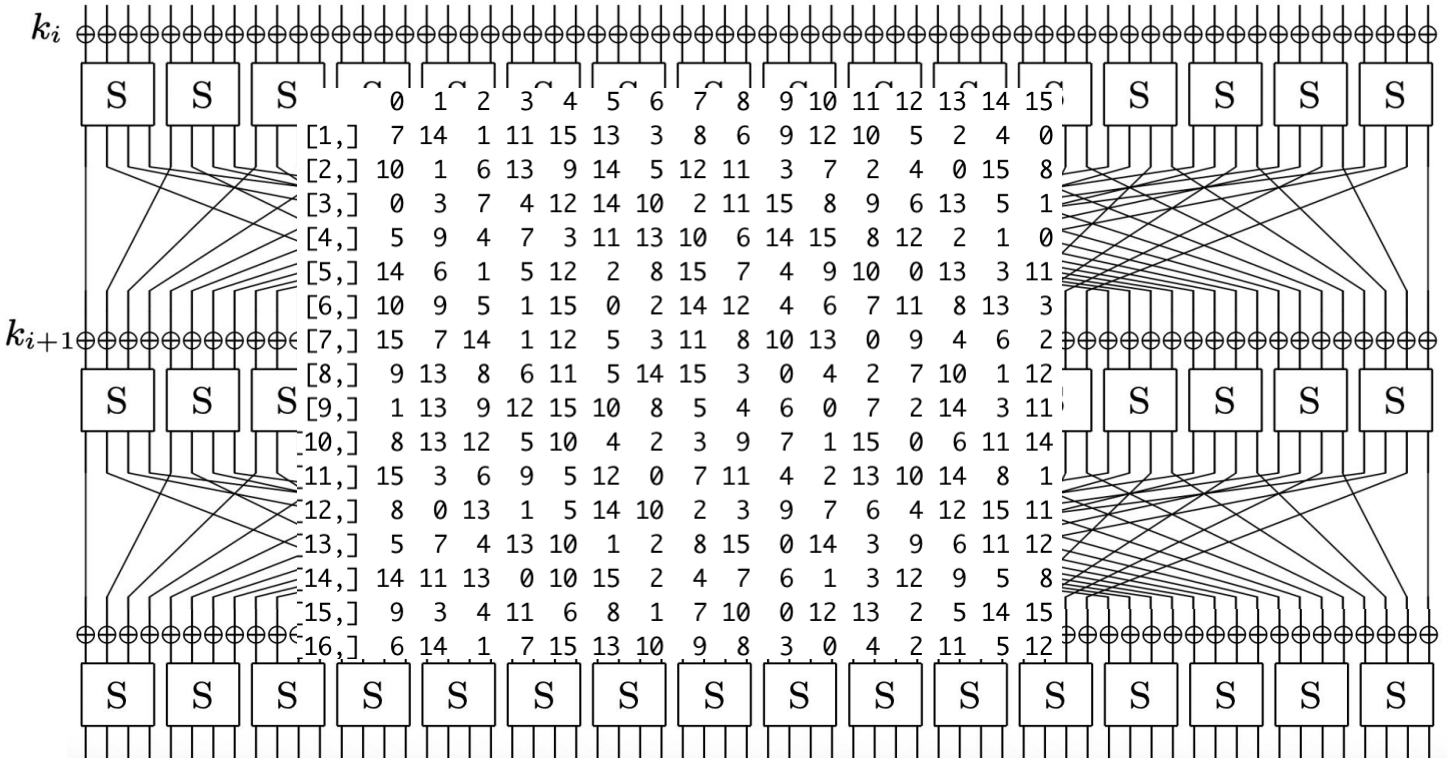
78 -> 4e→
00 01 01 01 00 00 01 00 ->
01110010 ->
0111 **0010**

1 XOR 1 = 0
1 XOR 0 = 1
0 XOR 1 = 1
0 XOR 0 = 0

Bogdanov, et al., 2007

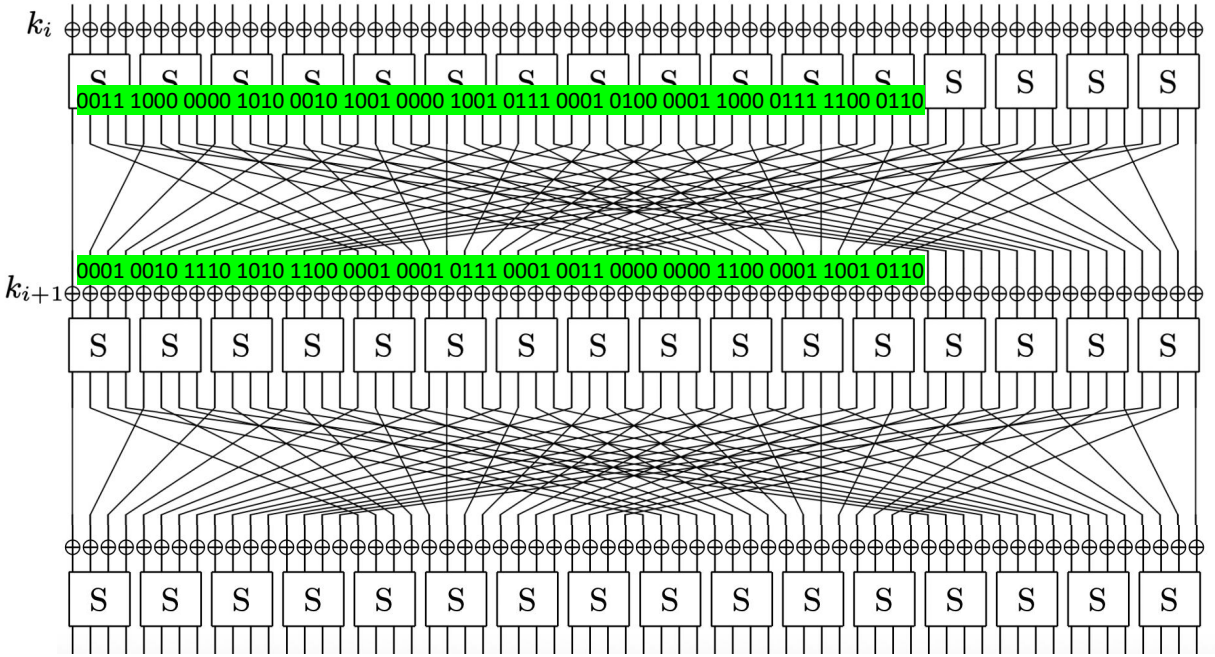
PRESENT – design zpracování bloku

zpráva 0101 1000 0000 0000 1001 1000 1101 0000 1011 0000 0101 0000 1010 0000 1000 0000
0011 1000 0000 1010 0010 1001 0000 1001 0111 0001 0100 0001 1000 0111 1100 0110



Bogdanov, et al., 2007

PRESENT – design zpracování bloku



- Jedno písmeno – 8 bitů.
- Na každý bit aplikovaný jiný bit z klíče.
- Na 4 první bity jedna substituce,
- na další 4 bity druhá substituce.
- Zamíchání bitů na jiné pozice tak, že po dalším míchání vstoupí vliv každého písmena do každého 4 bitu – tedy do každé půlky znaku (8 bit).

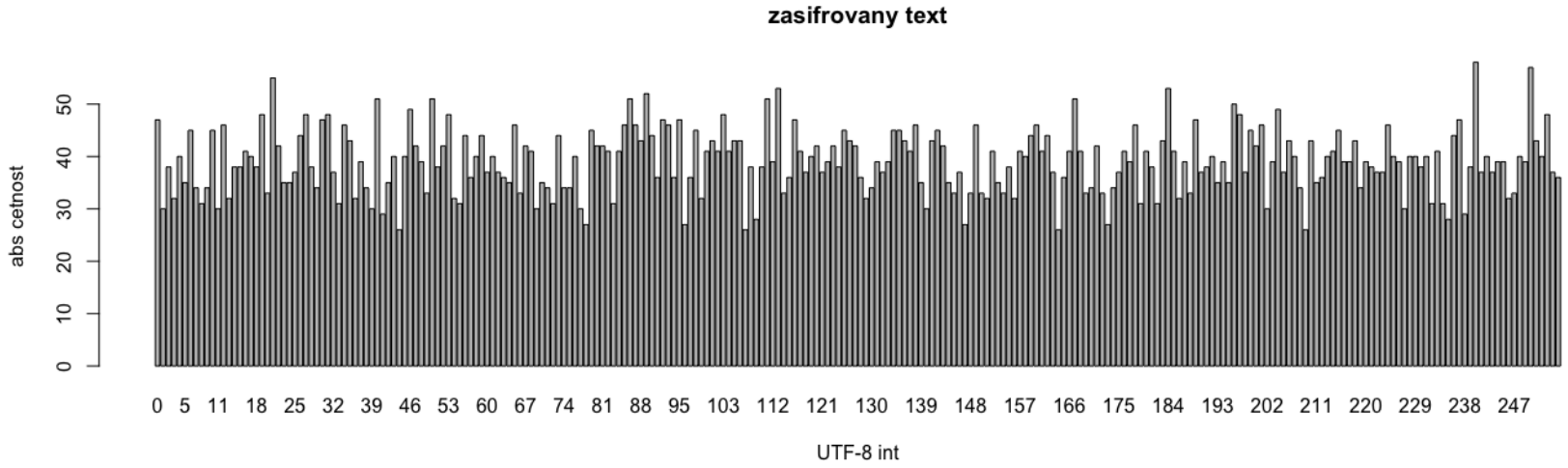
- Po každém kole se mění i klíč – rotuje doleva o 61 znaků, prvních 8 znaků se transformuje a poslední 3 a první 2 z následující části klíče provede xor s číslem kola / roundu.

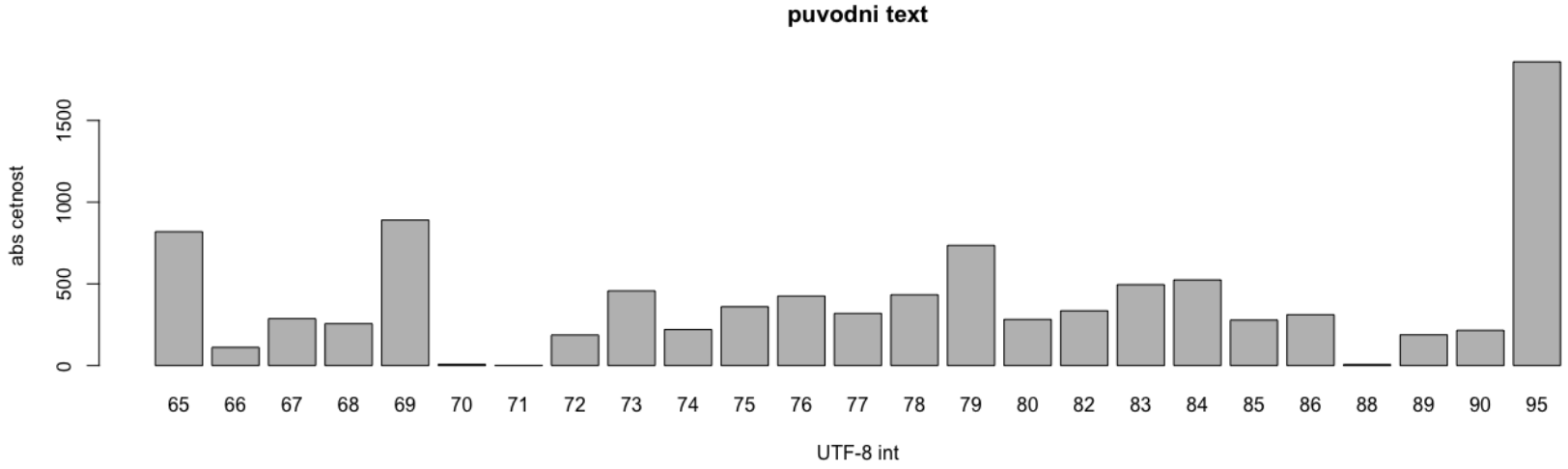
Bogdanov, et al., 2007

PRESENT

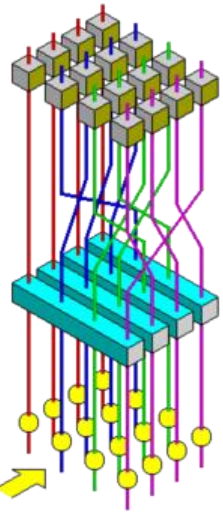
- V současné době je plně implementovaná šifra PRESENT na hranici prolomení (29 round už by bylo prolomitelné, ale ona má 32).
- Šifra poskytuje dobré vlastnosti a je „ultra lehká“ oproti, například AES.
- Šifra je citlivá na vstup a i na klíč – tedy neumožňuje odhalit klíč, když je znám jak nezašifrovaný, tak i zašifrovaný text.

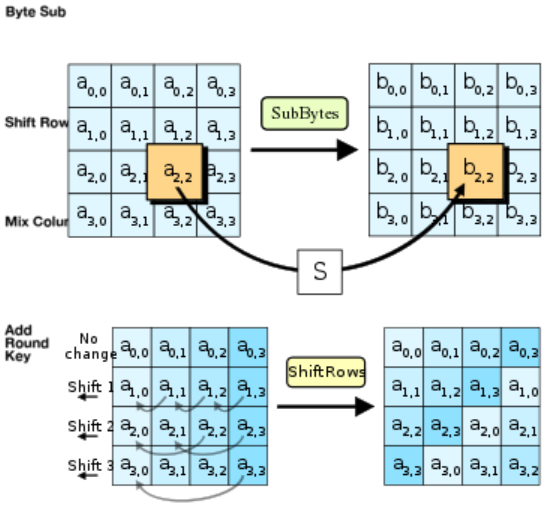
```
> #citlivost na zmenu hesla:
> present_enc("ABCDEFGHIJKLM","NASEHESLO")
 [1]  14  86 148 248 180  84 127 216 171 253  93 122  22  79 207  17
> present_enc("ABCDEFGHIJKLM","ABCDEFGHIJ")
 [1] 135 222  57 222  96 206  62  51 156 108 108  17 187   4 177  44
> #citlivost na zmenu textu:
> present_enc("ABCDEFGHIJKLM","NASEHESLO")
 [1]  14  86 148 248 180  84 127 216 156  30 181 107  24  84  35   6
> present_enc("BBCDEFGHIJKLM","NASEHESLO")
 [1]   1  68 142   6 177 153  74   4 151  38  97  69 105 194  65 227
```


zasifrovany text
abs cetnost
0
10
20
30
40
50
0 5 11 18 25 32 39 46 53 60 67 74 81 88 95 103 112 121 130 139 148 157 166 175 184 193 202 211 220 229 238 247
UTF-8 int

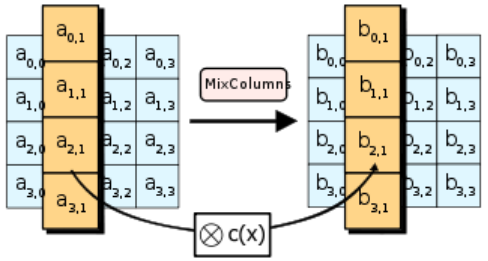

puvodni text
abs cetnost
0
500
1000
1500
65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 82 83 84 85 86 88 89 90 95
UTF-8 int

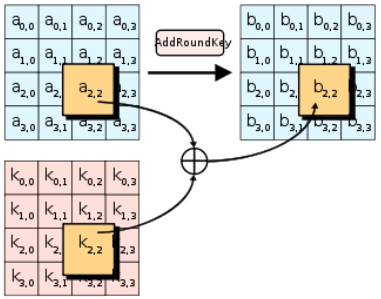
AES





- AES – 1990
- Advanced encryptin standard (AES)
- Varianta jiné blokové šifry (Rijndael)
- Princip:
  - 1 krok Expanze klíče na 128 bitů
  - 2. krok :Každý byt je prohnaný XOR operací (1xor1 = 0, 0xor0 = 0, 1xor0=1,0xor1=1)
    - Lze zpětně dekodovat!
  - 9-13kroků:
    - Transpozice řádek (proházení řádek)
    - Substituce bitů (podle pravidla, proházení pozic matic)
    - Transpozice sloupců (proházení sloupců)
    - Na každý bit je provedena xor s klíčem
  - 10-14 kol:
    - Substituce bitů
    - Transpozice řádek
    - Xor operace s klíčem





Zdroje obrázků: https://en.wikipedia.org/wiki/Advanced_Encryption_Standard

VŠE / FAKULTA INFORMATIKY A STATISTIKY

Karel Šafr, Ph.D.
karel.safr@vse.cz

Zdroje

Vysoká škola ekonomická v Praze

Zdroje

- Stamp M., IOW, R. M. 2007, Applied cryptanalysis: Breaking ciphers in the Real World, John Wiley & Sons inc..
- Helen Fouché Gaines, 1965. Cryptanalysis: a study of ciphers and their solution. Dover publications, Inc. New York
- Al Sweigart,2018 Cracking codes wiht python: An introduciton to Building and breaking ciphers,No strach press, San Francisco
- Easttom, W., 2021. Modern Cryptography: Applied mathematics for Encryption and Information security.Springer
- Bogdanov, A., Knudsen L.R., Leander, G., Paar, C., Poschmann A., Robshaw M.J.B., Seurin, Y., Vikkelsoe. C., 2007. PRESENT: An Ultra-Lightweight Block Cipher, https://www.iacr.org/archive/ches2007/47270450/47270450.pdf
- Bauer, F.L. 1998. Decrypted Secrets, Methods and Maxims of Cryptology, 4nd ed., Springer
- Rupa, Ch. Sirajuddin M. , 2020.. A Closer Look at Cybersecurity and Cryptanalysis Nova Science Publishers.
- Cryptography, 2020 WS2020/21. Institute of Applied Information Processing and Communications, Graz University Of Technology, https://online.tugraz.at/tug_online/wbLv.wbShowLVDetail?pStpSpNr=336732&pSpracheNr=2&pMUISuche=FALSE, popřípadě: https://www.iaik.tugraz.at/course/cryptography-705066-wintersemester-2022-23/
- Sinkov, A.,2009. Elementary Cryptanalysis: A Mathematical approach Mathematical Association of America, Anneli Lax New Mathematical Library, volume 22, 2nd ed., Revised by Todd Feil.
- Katz J., Lindell Y. 2021. Introduction to Modern Cryptography. CRC press, Taylor & Francis group.
- Joux, A., 2009. Algorithmic Cryptanalysis, Xchapman & Hall / CRC, Tylor and francis group.
- Paar C., Pelzl J. 2010. Understanding Cryptography: A textbook for students and practitioners. Springer.
- Nachef V. Patarin J., Volte E., 2017. Feistel Ciphers: Security proofs and cryptanalysis. Springer.
- Etienne, El. 2018-19. Eelementary Statistical methods of cryptography. Department of mathematics, Faculty of Sciences. Liege université
- Hwang S.O., Kim I. Lee W.K. 2021. Modern Cryptography with proof Techniques and Implementations.Taylor & Francis.
- Buell, D., 2021. Fundamentals of Cryptography: Introducting Mathematical and Algorithmic foundations, Springer & UTiCS
- Dooley, John F., 2018. History of Cryptography and Cryptanalysis: Codes, Ciphers, and Their Algorithms, Springer.
- Bauer, Craig. P., 2021 . Secret History: The story of Cryptology, 2nd ed. CRC press, Taylor & Francis Group
- Mihailescu M. I. Nita L. S. 2021. Cryptography and Cryptanalysis in MATLAB, APRESS.
- Wong, D., 2021. Real-World Cryptography, manning. Vol. 12.
- Friedman W., 1980. Military Cryptanalysis, Aegean Park Press

VŠE / FAKULTA INFORMATIKY A STATISTIKY

Jméno a příjmení autora
jmeno.prijmeni@vse.cz

Děkuji za pozornost

Vysoká škola ekonomická v Praze