Crypto-World

Informační sešit GCUCMP
ISSN 1801-2140

Ročník 16, číslo 6-7/2014 **24. červenec**

6 - 7/2014

Připravil: Mgr. Pavel Vondruška

Sešit je přednostně distribuován registrovaným čtenářům.
Starší sešity jsou dostupné na adrese
http://crypto-world.info/
(1394 registrovaných odběratelů)

Obsah :

Příloha: CFP_MKB2014.pdf http://crypto-world.info/casop16/CFP_MKB2014.pdf

A Identifikácia zmysluplného textu pri klasických šifrách

Patrik Matiaško, patrik.matiasko@gmail.com

ÚIM, FEI STU v Bratislave

Abstrakt

Zmyslom tejto práce je priblížiť niektoré z problémov identifikácie zmysluplného textu pri lúštení klasických šifier, a odporučiť možné postupy pri jeho hľadaní. Vezmeme si texty zašifrované troma šiframi. Konkrétne sa jedná o jednoduchú substitúciu[1], vigenèrovú šifru[2] a úplnú tabuľkovú transpozíciu[3]. Tieto šifry sú známe obmedzenou množinou kľúčov. Dajú sa overiť všetky kľúče a zostaviť tak množina kanidátov na OT. Potrebujeme však určiť ten správny OT spomedzi všetkých kandidátov. Tým pádom, ak to zhrnieme, cieľom tejto práce nie je vytvoriť program na lúštenie týchto šifier ale hodnotiť texty, na základe nami vytvorených ohodnocovacích funkcií a stanoviť možné postupy a odporúčania pri hľadaní zmysluplného textu.

Použité skratky

- CZ, DE, EN, SK – český, nemecký, anglický a slovenský jazyk
- TSA – štandardná telegrafná abeceda, 26 znakov bez diakritiky
- IC – index koincidencie
- OT – otvorený text
- ZT – zašifrovaný text

Použitý hardware

- Procesor : Intel(R) Core(TM) i5-3317U CPU @ 1.70Ghz
- Ram : 6,00GB
- OS : Windows 8.1

1 Skúmané texty

Na úplnom začiatku sme si zostavili 25 otvorených textov (teda nešifrované texty) v štyroch jazykoch. Texty sme vyberali zámerne odlišného žánru aby sme pokryli väčšiu množinu textov a zaručili tak ich rôznorodosť. Vybrane texty majú veľkosť od 600 znakov do 999 znakov. Je to z dôvodu zaručenia dostatočne dlhého textu, na ktorom sa bude dať využiť analýza či už frekvencie znakov alebo n-gramov. Texty sme zozbierali v štyroch jazykoch: CZ, DE, EN a SK. Samozrejmosťou bolo odstránenie diakritiky a špeciálnych znakov, teda prevedenie do TSA. Spôsob konverzie textov do TSA je popísaný v nasledujúcich tabuľkách. Anglický jazyk netreba nijako meniť nakoľko používa iba 26 znakov TSA.

CZ	ě/Ě	ř/Ř	ů / Ü
TSA CZ	E	R	U

DE	ä/Ä	ö/Ö	ß	ü/Ü
TSA DE	AE	OE	SS	UE

[1] http://en.wikipedia.org/wiki/Substitution_chiper

[2] http://en.wikipedia.org/wiki/Vigenere_cipher

[3] http://en.wikipedia.org/wiki/Transposition_cipher

SK	á/Á/ä	č/Č	ď/Ď	é/É	í/Í	Í/Ĺ/ľ/Ľ	ň/Ň
TSA SK	A	C	D	E	I	L	N
SK	ó/Ó/ô	ŕ/Ŕ	š/Š	ť/Ť	ú/Ú	ý/Ý	ž/Ž
TSA SK	O	R	S	T	U	Y	Z

Nasledovalo šifrovanie textov z otvorených na zašifrované texty. Celkovo sme tak dostali 300 zašifrovaných textov nakoľko používame tri typy šifier. Potom sme použili online generátor[4] na vygenerovanie náhodného kľúča dĺžky 8 znakov (RXLQEKNB) a zašifrovali sme niekoľko textov. Pri ich dešifrovaní a analýze sme narazili na veľkú časovú náročnosť výpočtov čo nepokrýval výkon nám dostupných počítačov.

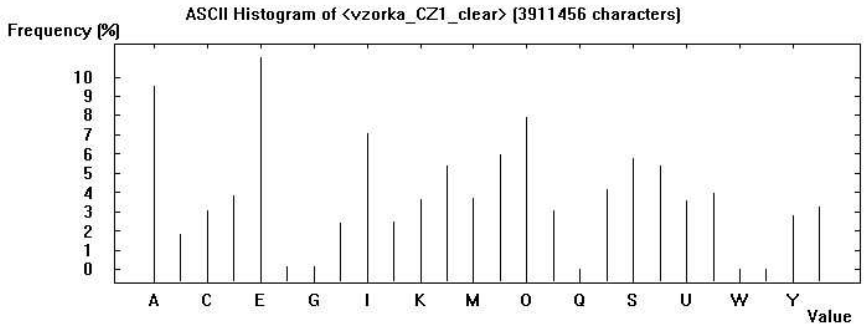
Po redukcií hesla na 5 znakov sme zvolili heslo: **JVLTN** ktoré sme použili na šifrovanie všetkých OT a získali sme tak všetky možné ZT. Pri tejto dĺžke hesla sme získavali výsledky v reálnom čase, a tak sme s tým mohli ďalej pracovať.

2 Použité štatistiky

V práci používame štatistiky ako sú histogram, rózne n-gramy, index koincidencie a viaceré kombinácie uvedených štatistík. Vzorové hodnoty uvedených štatistík pre jednotlivé jazyky sme získali z korpusov týchto jazykov. Spomínané štatistiky sme vybrali preto, že patria medzi základné metódy analýzy textu.

2.1 Histogram

Histogram[5] je rozdelenie početností znakov. Vykresľuje sa obvykle do stĺpcového grafu (alebo je písaný do tabuľky), ktorý je vhodný na vyjadrenie intervalového rozdelenia početností. Je zložený zo stĺpcov, ktoré reprezentujú jednotlivé intervaly. Intervalové hodnoty sa nanášajú na os x. Výška stĺpca zodpovedá hodnote početnosti intervalu pre konkrétne písmeno abecedy. Relatívna frekvencia sa vyznačuje na osi y [1]. Následné histogramy sú získané z korpusov pre uvedené jazyky:

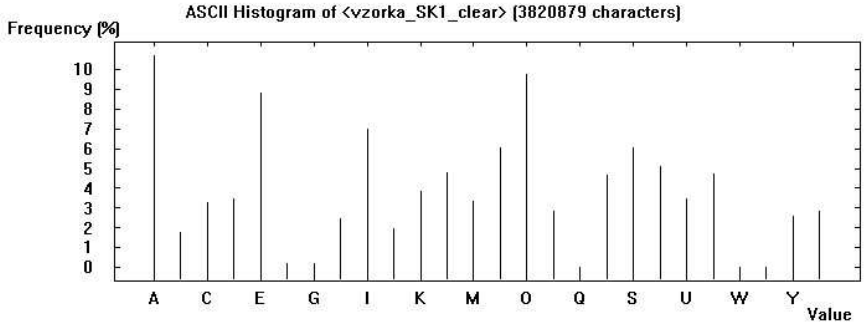


Obr. 1: CZ histogram

[4] http://www.random.org/strings/
[5] http://en.wikipedia.org/wiki/Histogram

A	B	C	D	E	F	G	H	I	J	K	L	M
9,75	1,82	3,12	3,87	10,99	0,15	0,17	2,38	6,73	2,42	3,68	5,46	3,71
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
5,95	7,93	3,03	0,01	4,11	5,59	5,37	3,51	3,94	0,01	0,02	2,75	3,36

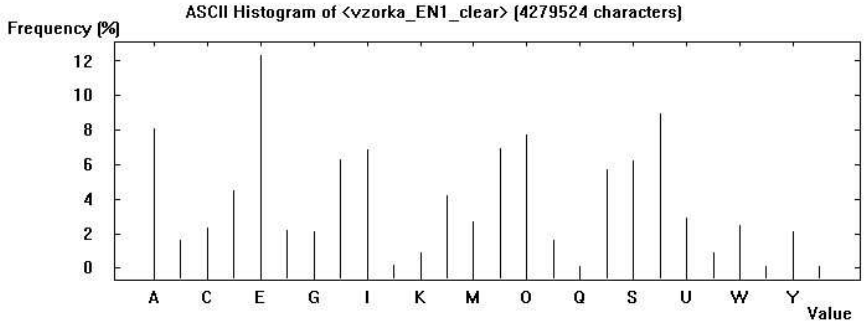
Tabuľka 1: Frekvencie znakov v CZ korpuse



Obr. 2: SK histogram

A	B	C	D	E	F	G	H	I	J	K	L	M
11,09	1,72	3,28	3,42	8,89	0,21	0,25	2,24	6,85	2,21	3,62	4,75	3,32
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
5,83	9,72	2,93	0,01	4,57	5,70	5,39	3,47	4,59	0,02	0,06	2,49	3,03

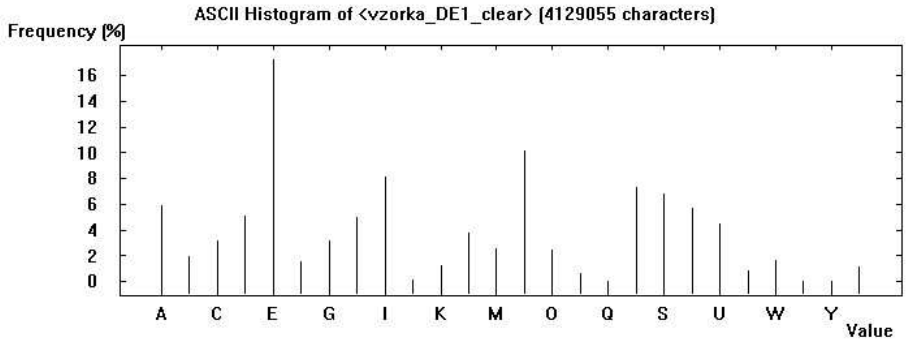
Tabuľka 2: Frekvencie znakov v SK korpuse



Obr. 3: EN histogram

A	B	C	D	E	F	G	H	I	J	K	L	M
7,87	1,81	2,55	4,18	11,87	2,25	2,34	6,14	6,80	0,20	1,02	4,68	2,64
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
6,74	7,70	1,91	0,11	5,97	6,57	8,36	2,82	0,82	2,18	1,12	2,05	0,09

Tabuľka 3: Frekvencie znakov v EN korpuse



Obr. 4: DE histogram

A	B	C	D	E	F	G	H	I	J	K	L	M
5,97	1,96	3,24	5,09	17,19	1,44	3,02	5,16	7,86	0,17	1,12	3,86	2,63
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
9,75	2,42	0,59	0,01	7,51	6,79	5,85	4,45	0,84	1,73	0,01	0,01	1,19

Tabuľka 4: Frekvencie znakov v DE korpuse

2.2 N-gramy

Najčastejšie bigramy, trigramy a tetragramy sme vypočítali frekvenčnou analýzou z korpusov a sú uvedené v nasledujúcich tabuľkách. Použili sme na to voľne dostupný program Cryptool[6].

Význam riadkov v tabuľkách:

- Riadok **BI**: 10 najfrekventovanejších bigramov.
- Riadok **TRI**: 10 najfrekventovanejších trigramov
- Riadok **TETRA**: 10 najfrekventovanejších tetragramov
- Riadok **%**: percentuálne zastúpenie bi-, tri-, tetragramov

Jazyk	CZ									
Poradie	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.
BI	NE	SE	NA	ST	AL	EN	LA	EM	PR	TE
%	1.8029	1.4328	1.3029	1.2956	1.2951	1.2879	1.2543	1.1554	1.0938	1.0596
TRI	SEM	PRO	STA	JSE	OST	OVA	EST	ENE	ALE	BYL
%	0.4138	0.3253	0.3238	0.3236	0.3131	0.3040	0.2856	0.2815	0.2757	0.2741
TETRA	JSEM	OVAL	KTER	PRAV	JAKO	SEMS	STAL	KDYZ	PRED	AJSE
%	0.3189	0.1392	0.1230	0.1295	0.1059	0.1048	0.0969	0.0931	0.0868	0.0862

Tabuľka 5: Najčastejšie n-gramy v CZ

[6] `http://www.cryptool.org/en/`

Jazyk	DE									
Poradie	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.
BI	ER	EN	CH	EI	DE	ND	TE	IN	IE	GE
%	3.8384	3.7219	3.0486	2.5373	2.2922	2.0729	1.9843	1.8113	1.5685	1.5625
TRI	EIN	ICH	DER	SCH	UND	NDE	DIE	CHE	INE	END
%	1.1874	1.1555	0.8924	0.8910	0.7799	0.7625	0.6789	0.6680	0.6480	0.5695
TETRA	EINE	CHEN	ICHT	NDER	SCHE	LICH	ENDE	NICH	SEIN	EICH
%	0.6105	0.3017	0.3000	0.2841	0.2651	0.2613	0.2507	0.2309	0.2272	0.2251

Tabuľka 6: Najčastejšie n-gramy v DE

Jazyk	EN									
Poradie	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.
BI	HE	TH	IN	ER	AN	ES	RE	ST	ND	ED
%	2.8740	2.8311	1.9393	1.7078	1.5112	1.2628	1.2013	1.1606	1.1217	1.1099
TRI	THE	ING	AND	HER	HIS	THA	HAT	ERE	NTH	ENT
%	1.1815	0.8126	0.7682	0.5322	0.3532	0.3440	0.3415	0.3370	0.3317	0.2875
TETRA	THER	NTHE	THAT	WITH	HERE	DTHE	OFTH	FTHE	THES	OTHE
%	0.2815	0.2738	0.2328	0.2232	0.1830	0.1763	0.1736	0.1678	0.1644	0.1637

Tabuľka 7: Najčastejšie n-gramy v EN

Jazyk	SK									
Poradie	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.
BI	ST	OV	NA	NE	AL	TO	EN	LA	RA	AN
%	1.4779	1.3732	1.3720	1.3666	1.1939	1.1244	1.0936	1.0888	1.0514	1.0279
TRI	OST	STA	OVA	EHO	PRE	PRI	YCH	TOR	EST	OVE
%	0.4251	0.3916	0.3825	0.3634	0.3509	0.3137	0.2918	0.2799	0.2611	0.2594
TETRA	KTOR	NOST	OSTA	SVOJ	OVAL	PRAV	JEHO	OSTI	PRED	STAV
%	0.2008	0.1274	0.1236	0.1181	0.1175	0.1140	0.1067	0.1061	0.0992	0.0922

Tabuľka 8: Najčastejšie n-gramy v SK

2.3 Možné kľúče

Pre začiatok sme si museli vygenerovať všetky možné kľúče ktoré pripadajú do úvahy pre jednoduchú substitúciu a Vigenèrovú šifru s dĺžkou hesla päť znakov. V jednoduchej substitúcií sa znaky v hesle nemôžu opakovať a teda počet možných hesiel je $26.25.24.23.22 = \mathbf{7\,893\,600}$. Ďalej toto heslo treba doplniť do dĺžky 26 znakov aby sa v ňom vyskytoval každý znak abecedy práve raz. Teda ak vygenerované heslo je: *FPQLC* tak doplnené heslo vyzerá nasledovne: *FPQLCABDEGHIJKMNORSTUVWXYZ*. Pri Vigenèrovej šifre sa znaky v hesle môžu opakovať čo znamená $26^5 = \mathbf{11\,881\,376}$ možných kľúčov.

Vygenerované heslá sme si uložili do textových súborov, aby sme s nimi mohli ďalej pohodlne pracovať a nemuseli ich opakovane generovať. To nám urýchľuje výpočtový čas.

Pri transpozíciach budeme využívať iné metódy detekcie zmysluplného textu, takže pre transpozíciu sme si možné kľúče generovať nemuseli. Text rozdelíme do *n* stĺpcov a získame tak možný otvorený text. Počet možností je *n!*, preto má zmysel uvažovať len do *n=13*, pretože už *číslo 13!* má dosť vysokú časovú náročnosť pri spracovaní na bežných počítačoch. Na novo zostavenom texte využívame naše vyhodnocovacie funkcie a určujeme tak správnu permutáciu stĺpcov. Ak by sme používali *n=13* museli by sme tak prejsť **227 020 800** možností. Vieme že bola použitá úplná transpozícia. Tým pádom musí byť veľkosť zašifrovaného textu deliteľná počtom stĺpcov ktoré chceme skúmať. Toto tvrdenie podkladáme nasledujúcou tabuľkou.

Názov	Počet znakov	1	2	3	4	5	6	7	8	9	10	11	12	13
CZ_01	680	X	X	-	X	X	-	-	X	-	X	-	-	-
CZ_02	905	X	-	-	-	X	-	-	-	-	-	-	-	-
CZ_03	710	X	X	-	-	X	-	-	-	-	X	-	-	-
DE_01	720	X	X	X	X	X	X	-	X	X	X	-	X	-
DE_02	730	X	X	-	-	X	-	-	-	-	X	-	-	-
DE_03	680	X	X	-	X	X	-	-	X	-	X	-	-	-
EN_01	745	X	-	-	-	X	-	-	-	-	-	-	-	-
EN_02	1000	X	X	-	X	X	-	-	X	-	X	-	-	-
EN_03	930	X	X	X	-	X	X	-	-	-	X	-	-	-
SK_01	720	X	X	X	X	X	X	-	X	X	X	-	X	-
SK_02	685	X	X	-	-	X	-	-	-	-	-	-	-	-
SK_03	615	X	-	X	-	X	-	-	-	-	-	-	-	-

Tabuľka 9: Redukcia možností stĺpcov v transpozíciách

Veľké *X* znamená že veľkosť daného textu je deliteľná príslušným počtom stĺpcov bez zvyšku. Pomlčka označuje, že počet znakov daného textu nie je deliteľný veľkosťou pre príšlušný počet stĺpcov. Jediný stĺpec ktorý nám túto podmienku spĺňa, je práve stĺpec 5 a teda keď rozdelíme text do stĺpcov dĺžky päť, nezostane nám v tabuľke žiadne prázdne políčko. Samozrejme aj počet stĺpcov jeden je v každom prípade jedna z možností. Aj stĺpec pre 10 stĺpcov sa vyskytuje dosť často, avšak to by nám skomplikovalo situáciu nakoľko by sme museli uvažovať o 10! možných textov čo by oveľa viac zvýšilo čas výpočtu. Z uvedenej tabuľky sme sa preto rozhodoli pre počet stĺpcov päť.

3 Ohodnocovacie funkcie

V práci využívame ohodnocovacie funkcie pre hodnotenie kandidátov na zmysluplný text, na základne vlastností n-gramov, indexu koincidencia a kombinácií uvedených štatistík. Maximálne n-gramy, ktoré využívame sú tetragramy. Použitie vyšších n-gramov je výpočtovo aj časovo náročné a neprináša adekvátne zlepšenie výsledkov.

3.1 Histogramy

Histogramové štatistiky sme vytvorili na základe zozbieraných korpusov. Vytvorili sme si program, ktorý si vypočíta štatistiku z priloženého korpusu a následne ju porovnáva so štatistikou, ktorú si program určí zo ZT. V programe si môžeme

zvoliť, koľko najlepších výsledkov si chceme uložiť, nepotrebujeme všetky ale len X najlepších. Pridali sme aj možnosť výberu akým bude program vyhodnocovať vzorky. Sú aplikované 2 metódy, konkrétne geometrická vzdialenosť a absolútna vzdialenosť. Pri testoch sme využívali kritérium geometrickej vzdialenosti. Porovnávali sme ideálne vzdialenosti vo vektore zostaveného z korpusu, oproti vektoru zostaveného z možného kľúča. Keďže, v histograme je len dvadsaťšesť znakov netreba výsledky redukovať o nulové histogramy, pretože také v korpuse nie sú. Tento parameter budeme využívať pri vyšších n-gramoch.

3.1.1 Použitie histogramov

Do nasledujúcich tabuliek sme zapísali priemerné poradie kľúča, ktoré sme určili ako aritmetický priemer poradí správneho kľúča vo všetkých vzorkách textov.

Jazyk	Substitúcia	Vigenère
CZ	78,2	15,8
DE	21	1
EN	31,8	1,6
SK	18,4	5

Tabuľka 10: Výsledky funkcie *histogram*

V tejto tabuľke uvádzame len 2 použité šifry. Transpozičná šifra sa tu nenachádza, pretože pri tomto type šifry nemá funkcia histogram žiaden význam. Ak vychádzame z poznatkov o klasických šifrách, vieme, že transpozičné šifry zachovávajú frekvencie znakov [1].

3.2 Bigramy

Celkový počet bigramov je $26 * 26 = 676$. V programe si môžme svoje výpočty urýchliť tým, že neberieme do úvahy celý vektor o dĺžke 676, ale vezmeme len časť bigramov. Toto sme ošetrili tým, že ak sa nejaký bigram v korpuse nenachádza, skrátime tak svoj vektor, s ktorým porovnávame ideálny vektor. Pri pokusoch sme prišli na to, že takto skrátime vektor v priemere o 7%. Ďalšie urýchlenie prebieha v podobe vloženia konštanty do programu, ktorá určí aký dlhý vektor sa berie do porovnávania a teda uvažujeme len s n najvýznamnejšími bigramami, kde počet n si definujeme podľa potreby. Táto metóda dosahuje lepší výpočtový čas, zároveň však horšie výsledky. Treba presne zvážiť, na koľko sa má skrátiť vektor, aby dosahoval ešte zmysluplné výsledky.

3.2.1 Použitie bigramov

Význam stĺpcov tabuľky 11: **Jazyk** – jeden z vybraných jazykov, **Substitúcia, Vigenère, Transpozícia** – typy šifier a **Dáta v tabuľke** – priemerné poradie výskytu zmysluplného textu.

V pokusoch sme skracovali vektor bigramov na rôzne dĺžky a hľadali sme tak optimálnu dĺžku vektora na porovnávanie z hľadiska rýchlosti a správnosti výpočtu.

Do tabuľky sme uviedli priemerné poradie kľúča, ktoré závisí od dĺžky porovnávaného vektora presnejšie od počtu použitých najvýznamnejších bigramov.

Jazyk	Substitúcia	Vigenère	Transpozícia
CZ	1,4	10,4	55,6
DE	1	1	42,3
EN	1,4	1,2	48,2
SK	1,2	32,4	58,3

Tabuľka 11: Výsledky funkcie *bigram*

Jazyk	626	600	550	500	400	300	200	100
CZ	1,4	1,4	1,4	1,4	1,4	1,6	2,3	3,9
DE	1	1	1,2	1,2	1,2	1,3	2,6	3,1
EN	1,4	1,4	1,5	1,5	1,5	1,5	2,1	3,7
SK	1,2	1,2	1,4	1,4	1,5	1,5	2,4	4,5

Tabuľka 12: Výsledky funkcie *bigram* so skráteným vektorom

Tento príklad je demonštrovaný na jednoduchej substitúcií. Z uvedených hodnôt vyplýva že najlepšie výsledky dosahujeme pri plnej dĺžke bigramového vektora a najrýchlejšie výsledky zasa pri najmenšej dĺžke vektora, čo sa dalo očakávať. Z dosiahnutých výsledkov je zrejmé, že k markantnému zhoršeniu výsledkov prichádza až keď použijeme menej ako 200 bigramov. Týmto krokom ušetríme niekoľko sekúnd až minút v závislosti od počítača, na ktorom robíme pokusy. Z našich pokusov rozdiel času medzi vektormi dĺžky 626 a 100 bol časový rozdiel 278 sekúnd. Rozhodli sme sa, že budeme využívať zrýchlenie iba odstránením nadbytočných nulových n-gramov, aby sme zachovali čo najpresnejšie výsledky.

3.3 Trigramy

Pri trigramoch je situácia už komplikovanejšia, pretože všetkých možných trigramov je $26^3 = 17\,576$. Samozrejme sme aplikovali vylepšenie ktoré redukuje nulové trigrami, ktoré nie sú v korpuse. Takto dokážeme vektor všetkých trigramov zredukovať na zhruba 14 000 čo predstavuje skrátenie o viac než 21% trigramov. Aj táto funkcia sa dá ešte manuálne upraviť čo sa týka dĺžky porovnávaného vektora. Na pokrytie 90% trigramov potrebujeme použiť 2 453 pri CZ, 1 522 pri DE, 2 630 pri EN a 2 648 trigramov pri SK. Tieto hodnoty sme zostavili z použitého korpusu. Pokrytie 95% trigramov nám zabezpečí použitie 3 294 trigramov pri CZ, 2 173 pri DE, 3 533 pri EN a 3 294 trigramov pri SK.

3.3.1 Použitie trigramov

Význam stĺpcov tabuľky 13: **Jazyk** – jeden z vybraných jazykov, **Substitúcia, Vigenère** – typy šifier a **Dáta v tabuľke**, priemerné poradie výskytu zmysluplného textu.

3.4 Tetragramy

Tetragramov je $26^4 = 456\,976$ a ich použitie je časovo ešte náročnejšie. Preto využívame redukciu nulových tetragramov, čím zredukujeme vektor o 17% tetragramov čo je v porovnaní s celkovým počtom malé číslo. Preto na tomto mieste

Jazyk	Substitúcia	Vigenère	Transpozícia
CZ	1,11	1,2	41,2
DE	1	1	38,8
EN	1	1	38,2
SK	1,1	1,3	49,9

Tabuľka 13: Výsledky funkcie *trigramy*

veľakrát používame práve výber X najvýznamnejších tetragramov na zrýchlenie výpočtov programu. V programe ich samotné využívame len pri transpozícií. Na pokrytie 90% tetragramov potrebujeme použiť 17 103 pri CZ, 9 755 pri DE, 16 193 pri EN a 18 802 tetragramov pri SK. Tieto hodnoty sme zostavili z použitého korpusu. Pokrytie 95% tetragramov nám zabezpečí použitie 23 618 tetragramov pri CZ, 11 272 pri DE, 19 214 pri EN a 26 068 tetragramov pri SK.

Jazyk	Transpozícia
CZ	4,7
DE	7,2
EN	6,6
SK	5,1

Tabuľka 14: Výsledky funkcie *tetragram*

Význam stĺpcov tabuľky 14: **Jazyk** – jeden z vybraných jazykov, **Substitúcia, Vigenère** – typy šifier a **Dáta v tabuľke**, priemerné poradie výskytu zmysluplného textu.

3.5 Index Koincidencie

Predstavuje pravdepodobnosť že dve náhodné zvolené písmena z textu sú zhodné. Vzorec pre výpočet indexu koincidencie je nasledovný:

$$IC = \sum_{i=1}^{c} \frac{n_i(n_i - 1)}{N(N-1)},$$

kde n_i je počet výskytov i-teho znaku abecedy v skúmanej vzorke, N je celkový počet znakov skúmanej vzorky a c je počet znakov použitej abecedy. V našom prípade $c = 26$, keďže používame TSA [3].

Pre jednoduchú substitúciu index koincidencie nezohráva významnejšiu úlohu. Môžeme ním zistiť maximálne pravdepodobnú dĺžku kľúča, ale nakoľko tento fakt poznáme tak nám nijako nepomôže pri identifikácií zmysluplného textu. Pri dešifrovaní správy daným kľúčom sa totiž nemení rozdelenie relatívnych početností znakov použitej abecedy, preto šifrovanie neovplyvní hodnotu indexu koincidencie skúmanej vzorky.

3.5.1 Použitie IC na Vigenèrovej šifre

V tabuľke 15 sú možné kľúče, ktoré sú adeptmi na zmysluplný text. Hodnoty sme získali tak, že sme poronávali Index koincidencie ZT s Indexom koincidenencie OT. Zhodné výsledky ukladali pre analýzu. Vybrali sme ich nezávislé od jazyka,

AMCKE	BNDLF	COEMG	DPFNH	EQGOI
FRHPJ	GSIQK	HTJRL	IUKSM	JVLTN
KWMUO	LXNVP	MYOWQ	NZPXR	OAQYS
PBRZT	QCSAU	RDTBV	SEUCW	TFVDX
UGWEY	VHXFZ	WIYGA	XJZHB	YKAIC
ZLBJD				

Tabuľka 15: Výsledky *ic* funkcie pre Vigenèrovú šifru

v ktorom bol samotný text, pretože pri všetkých jazykoch sme dostávali rovnaké výsledky ako sú uvedené v tabuľke. Môžme si všimnúť nasledovné súvislosti medzi jednotlivými adeptmi. Posun medzi *A,M* je 12 znakov. Tento posun platí v každej bunke tabuľky. Posun *B,N* je 12 atď. Posun medzi *M,C* je *1*6 znakov, samozrejme pracujeme s modulom 26, teda nemôžme presiahnuť TSA. Takto podobne by sme mohli pokračovať v opisovaní každej bunky. IC funkcia pre Vigenèrovú šifru nám teda udáva, že ak zvolíme prvé náhodné písmeno, ktoré je kandidátom na kľúč, napríklad *L* nasledujúce písmeno hesla musí byť posunuté o 12, 12+12=24, dostávame *X*. Takto dokážeme zostaviť celý kľúč. Posuny sú nasledovné: +12, +16, +8, +20. Samozrejme tieto posuny môžeme brať aj opačným smerom a získame takéto hodnoty: -14, -10, -16, -6 modulo 26.

Frekvenčnú analýzu a ani samotný index koincidencie nemá význam používať na akejkoľvek transpozičnej šifre, pretože transpozície zachovávajú pôvodné znaky. Preto by nám IC pri identifikácií zmysluplného textu nijako nepomohlo.

3.6 Kombinácie funkcií

Pri skúmaní možností riešení a vyhodnocovacích funkcií sme dospeli k názoru, že by bolo vhodné vyhodnocovacie funkcie medzi sebou kombinovať. Funkcie by si predávali výsledky jedna druhej, a tak by sme získavali stále lepších a lepších kandidátov na zmysluplný text. Toto je však ideálny prípad. V praxi to vyzerá tak že na rozšifrovaný ZT pri použití náhodného kľúča aplikujeme *funkciu1* a výsledných možných kandidátov znovu preusporiadame alebo zredukujeme použitím *funkcie2*. Vždy začíname s funkciou, ktorá je časovo a výpočtovo menej náročná a prechádzame k tým časovo náročnejším.

3.6.1 Histogram-Bigram

Kombinácia týchto dvoch funkcií priniesla zaujímavé výsledky. Ako prvú funkciu ktorú sme aplikovali na ZT je funkcia *histogram* a uložili sme si najlepších 10 000 výsledkov. Tieto výsledky sme použili ako vstupné kľúče do funkcie *bigramy*. Nemusíme teda dešifrovať 7 893 600 textov, pri jednoduchej substitúcií s použitím hesla dĺžky päť znakov, ale len 10 000 najlepších kandidátov na zmysluplný text, čo predstavuje redukciu skúšaných možností na 0,12% možností. Toto číslo sme zvolili preto, aby sme mali istotu že správny kľúč bude práve medzi vybranými 10 000 kandidátmi. Vychádzali sme z pokusov pri funkcií *histogram*. Tieto hodnoty sú deklarované v tabuľke pre funkciu *histogram*. Vidíme, že už pri jednoduchej substitúcií by nám stačilo priemerné 37,35 pokusov, a pri Vigenèrovej šifre nám stačí iba 5,85 pokusov. Výpočtový čas funkcie *histogram-bigram* je nižší ako v samotnej funkcií *bigramy*, pretože sme znížili počet možností o 99,88%. Čas výpočtu tejto kombinácie je o 70% menší ako pri samotnej funckie *bigram*.

Jazyk	Substitúcia	Vigenère
CZ	1,2	16,5
DE	1	1
EN	1	1,1
SK	1	18,1

Tabuľka 16: Výsledky funkcie *histogram-bigram*

Do tabuľky sme zapísali priemerné poradie kľúča, ktoré sme určili ako aritmetický priemer poradí správneho kľúča vo všetkých vzorkách textov.

3.6.2 Histogram-Bigram-Trigram

Kombinácia troch vyhodnocovacích funkcií ktoré si posúvajú výsledky a tým dochádzka k vylepšovaniu výsledkov. Pri funkcii *histogram* začíname s plným kľúčom. Z tejto funkcie si vyberáme 10 000 najlepších kandidátov na zmysluplný text a tieto pošleme do funkcie *bigram*, ktorá vráti 5 000 najlepších kandidátov, a tie sa posunú do funkcie *trigram*, ktorá vráti už len 1 000 najlepších. Samozrejme, zoradené od najlepšieho po najhorší. Takýmito krokmi sa časová náročnosť výpočtu skracuje a výsledky sú preto rýchlejšie dosiahnuteľné. Výsledky sú prezentované v nasledujúcej tabuľke. Výpočtový čas funkcie je o 80% nižší než v samotnej funkcii *trigram*.

Jazyk	Substitúcia	Vigenère
CZ	1,1	2,4
DE	1	1
EN	1	1,3
SK	1	4,1

Tabuľka 17: Výsledky funkcie *histogram-bigram-trigram*

Do tabuľky sme zapísali priemerné poradie kľúča, ktoré sme určili ako aritmetický priemer poradí správneho kľúča vo všetkých vzorkách textov.

3.6.3 IC-Histogram

ic v kombinácií s funkciou *histogram* sme používali iba pri Vigenèrovej šifre kde má funkcia *ic* význam a dosahuje tam pozitívne výsledky. Po funkcii *ic* sme zredukovali možnosti pre zmysluplný text na 26 a na túto množinu sme aplikovali najrýchlejšiu funkciu *histogram*, ktorá v tomto prípade plne postačuje. Aplikácia inej funkcie už nemá význam, pretože výsledky tejto kombinácie sú postačujúce. Zámenou funkcie *histogram* za inú, napríklad *bigram*, by sme predĺžili výpočtový čas o 200% a výsledky funkcie by boli rovnaké.
Do tabuľky sme zapísali priemerné poradie kľúča, ktoré sme určili ako aritmetický priemer poradí správneho kľúča vo všetkých vzorkách textov.

3.6.4 Histogram-Bigram-Trigram-Tetragram

Táto kombinácia všetkých významných N gramov nám zaručuje dobré výsledky. Začíname s plným kľúčom a po funkcii *histogram* dostávame už len 10 000 výsledkov. Pokračujeme funkciou *bigram*, ktorá výsledky redukuje na 5 000 kandidátov

Jazyk	Vigenère
CZ	1
DE	1
EN	1
SK	1

Tabuľka 18: Výsledky funkcie IC-Histogram

následne funkcia *trigram*, ktorá výsledky redukuje na 1 000 výsledkov a nakoniec to zavŕšime funkciou *tetragram*. Získame tak najlepších 500 výsledkov. Takoutou kombináciou funcií ušetríme 90% času oproti samotnej funkcii *tetragram*.

Jazyk	Substitúcia	Vigenère
CZ	1	1,1
DE	1	1
EN	2,1	1
SK	1,1	1

Tabuľka 19: Výsledky funkcie *histogram-bigram-trigram-tetragram*

Do tabuľky sme zapísali priemerné poradie kľúča, ktoré sme určili ako aritmetický priemer poradí správneho kľúča vo všetkých vzorkách textov.

3.6.5 Histogram-Trigram

Touto kombináciou chceme poukázať na to, že ak vynecháme funkciu *bigram* tak nezískame markantne lepšie výsledky, len čas výpočtov sa predĺži ak pošleme do funkcie *trigram* väčší počet možností. Jediný rozdiel je v čase výpočtu, ktorý sa predĺži o viac než 70%.

Jazyk	Substitúcia	Vigenère
CZ	1,1	2,4
DE	1	1,1
EN	1	1,3
SK	1,2	4,2

Tabuľka 20: Výsledky funkcie *histogram-trigram*

Do tabuľky sme zapísali priemerné poradie kľúča, ktoré sme určili ako aritmetický priemer poradí správneho kľúča vo všetkých vzorkách textov.

4 Zhodnotenie výsledkov

Vo všeobecnosti neexistuje jediná univerzálna správna metóda, ktorou postupovať pri danej šifre a jazyku. Cieľom našej práce bolo nájsť niektoré z možností a odporučiť ich. Výsledky sme rozdelili podľa použitých šifier.

4.1 Odporúčané postupy

Funkcia je úspešná ak jej výsledok je vo vopred stanovenej množine, presnejšie, ak sa nachádza medzi prvými desiatimi najlepšími kandidátmi na zmysluplný text.

Tým sme splnili náš cieľ. Výsledok nemusí byť prvý, môže sa vyskytovať napríklad aj na štvrtom mieste. Všetko záleží od vstupného textu a jeho špecifikácií.

4.1.1 Jednoduchá substitúcia

Ako prvú možnú variantu uvádzame funkciu *bigram*, ktorá má o niečo lepšie výsledky ako funkcia *histogram-bigram*. Prvá uvedená má však vyššiu časovú náročnosť. Samotná funkcia *bigram* napĺňa cieľ tejto práce odhaliť zmysluplný text.

Použitím funkcie *histogram-bigram-trigram* sa výsledky dajú ešte viac spresniť avšak na úkor výpočtového času. Ak by sme použili aj kombináciu s *tetragramom*, tak dosiahnuté výsledky ešte vylepšíme, až na EN, v ktorom nastalo zhoršenie výsledkov. Toto mohlo byť spôsobené nevhodným výberom jazykového korpusu. Ak by sme tento korpus rozšírili, je možné, že by sa výsledky zlepšili.

4.1.2 Vigenèrová šifra

Na tento typ šifry odporúčame kombináciu funkcií *ic-histogram*, čo je najrýchlejšia a najúspešnejšia metóda, ktorú sme našli. Nakoľko po funkcií *ic* máme už len 26 adeptov na zmysluplný text, zoradíme ich následne funkciou *histogram*, ktorá je z n-gramov najrýchlejšia a to bez ohľadu na jazyk, v ktorom sa daný text nachádza.

Taktiež metóda ktorá prináša dobré výsledky je kombinácia 4 funkcií: *histogram*, *bigram*, *trigram* a *tetragram*, kde vieme takmer so 100% úspešnosťou nájsť zmysluplný text. Avšak táto metóda je niekoľkonásobne časovo náročnejšia ako *ic-histogram*. Taktiež funckia *histogram-bigram* prináša dobré výsledky pri použití na nemeckom a anglickom jazyku. Pre český a slovenský jazyk to však neplatí.

4.1.3 Úplná transpozícia

Jediná možnosť ktorá vyplýva z našich pokusov je použitie funkcie *tetragram*. Jedine táto funkcia spĺňa vopred stanovené ciele a to, že nájdeme zmysluplný text so stanovenou úspešnosťou. Je spomedzi všetkých funkcií najnáročnejšia na výpočet či už z hľadiska výkonu počítača alebo času. Pri použití funkcie *trigram* nedosahujeme tak presné výsledky ako by sme potrebovali. Ale opäť sa tu dá uplatniť vylepšenie v podobe výberu lepších textov na praktizovanie testu. Alebo aj rozšírenie korpusov o lepšie texty, v ktorých by bolo lepšie zastúpenie n-gramov.

Literatúra

[1] Otokar Grošek, Milan Vojvoda, Pavol Zajac: *Klasické šifry*, STU v Bratislave, 2007

[2] Jiří Janeček: *Odhalená tajemství šifrovacích klíčů minulosti*, Naše Vojsko, Praha 1994

[3] Friedman William F.: *The index of coincidence and its applications in cryptanalysis*, EAGEAN PARK PRESS, California 1989

B. Cryptorbit, 1.díl

RNDr. Vlastimil Klíma, nezávislý kryptolog – konzultant v.klima@volny.cz
Ing. Martin Kákona, martin.kakona@astro.cz

Abstrakt

V tomto článku popisujeme velmi nedávno detekovaný kryptovirus Cryptorbit a pokusíme se buď navrhnout algoritmus pro zpětné rozšifrování dat, zašifrovaných tímto virem, (pokud se nám podaří něco takového vymyslet) nebo alespoň motivovat ostatní k luštění. V době psaní článku žádnou úplnou metodu nemáme ani neznáme.

S příchodem anonymních a pseudonymních měn jako je například Bitcoin se rozšířila nová móda mezi viry, takzvaný „ransomware“. Jde o malware, který zabraňuje nějakým způsobem uživateli přístup k počítači. S oblibou se k tomuto účelu využívá kryptografie. Vir zašifruje uživateli soubory s daty a následně zobrazí informaci o tom, jak za úplatu získat klíč pro jejich dešifrování.

Tato myšlenka není nová. Historicky první ransomware byl zřejmě PC Cyborg již v roce 1989. Od té doby vznikla celá řada „vyděračského software“ založeného na kryptografii. Zajímavé také je, že většina autorů takovýchto virů byla dopadena. Dnes je nové to, že anonymní placení umožňuje autorům vyhýbat se zodpovědnosti za svoje činy. Respektive, například v případě Bitcoinu, je platba vystopovatelná, ale Interpol má zřejmě spoustu jiné práce. Uživatelům počítačů dnes tedy zbývají pouze dvě možnosti: zálohovat nebo se spolehnout na své znalosti v oboru kryptografie ;]

Nyní k samotnému Cryptorbitu. K datu publikace tohoto článku není známo, jak se Cryptorbit do počítače dostává, ale s největší pravděpodobností ho uživatel sám spustí. Také nikde neexistuje jeho vzorek, protože se ihned po zašifrování uživatelských dat z počítače sám smaže. V důsledku toho na něj nereagují antivirové programy.

Vir je navržen tak, aby zašifroval v co možná nejkratším čase co možná největší množství souborů s daty. Přitom se vyhýbá souborům operačního systému a spustitelným programům. Je to logické, protože operační systém bude uživatel ještě potřebovat pro provedení elektronické platby ;)

K dnešnímu dni existují dvě verze Cryptorbitu. První verze pro nás není zajímavá z toho důvodu, že používá příliš krátký klíč, takže není odolná proti útoku hrubou silou.
Druhá verze je kryptograficky zajímavější a mnohem nebezpečnější. Klíč, který se uživateli v současnosti prodává za 1BC[1], má délku 128 bitů.

[1] BC je zkratka pro bitcoin. Je to virtuální měna, která je založená na kryptografii, její použití pro obchodování, rychlé zbohatnátí nebo zchudnutí je kontroverzní a v současné době značně rizikové, nicméně je to velice velice zajímavý koncept.

Cena klíče se přitom může do budoucna měnit. Uživatel se cenu dozví teprve po zadání sériového čísla jeho verze Cryptorbitu na webový server, který je provozován na distribuované síti Tor[2].

Cryptorbit v.2 šifruje vždy prvních 1024 a posledních 1024 bajtů souboru (tyto bloky můžeme nazývat sektory).

Nejprve je vypočítán určitým způsobem 128bitový klíč, a z něho poté pracovní klíč. Oba výpočty ponechme prozatím stranou, neboť důležitý je nyní pouze pracovní klíč. Podstatné pro luštění tohoto klíče je, že tentýž pracovní klíč je platný pro celý počítač a použije se k šifrování všech souborů.

Pracovní klíč tvoří:

- tabulka 1024 bajtů N3[0], ..., N3[1023],
- čtyři 32bitové hodnoty **N1**, **N2**, **N4**, **N5**.

Ještě poznamenejme, že 32bitové hodnoty budeme označovat tučným písmem, ostatní jsou bajty. Pracovní klíč je tedy stejný pro jakýkoliv soubor, šifrovaný Cryptorbitem na daném počítači. Budeme se ho snažit určit z několika dvojic otevřeného a zašifrovaného tvaru sektorů 1024 bajtů. Proto bude nutné se snažit na počítači, napadeném Cryptorbitem, najít co nejvíce takových souborů, kde známe otevřený text buď ze zálohy nebo z operačního systému nebo aplikací.

Řekněme, že budeme analyzovat jeden zašifrovaný blok. Máme tedy 1024 bajtů CT[0], ..., CT[1023] zašifrovaného sektoru a 1024 bajtů otevřeného sektoru PT[0], ..., PT[1023]. Čtenáři Crypto-worldu jsou zvyklí, že PT je zkratka pro plaintext a CT pro ciphertext.

Bylo by báječné, kdyby Cryptorbit šifroval tak, že na otevřený text naxoruje klíč N3:

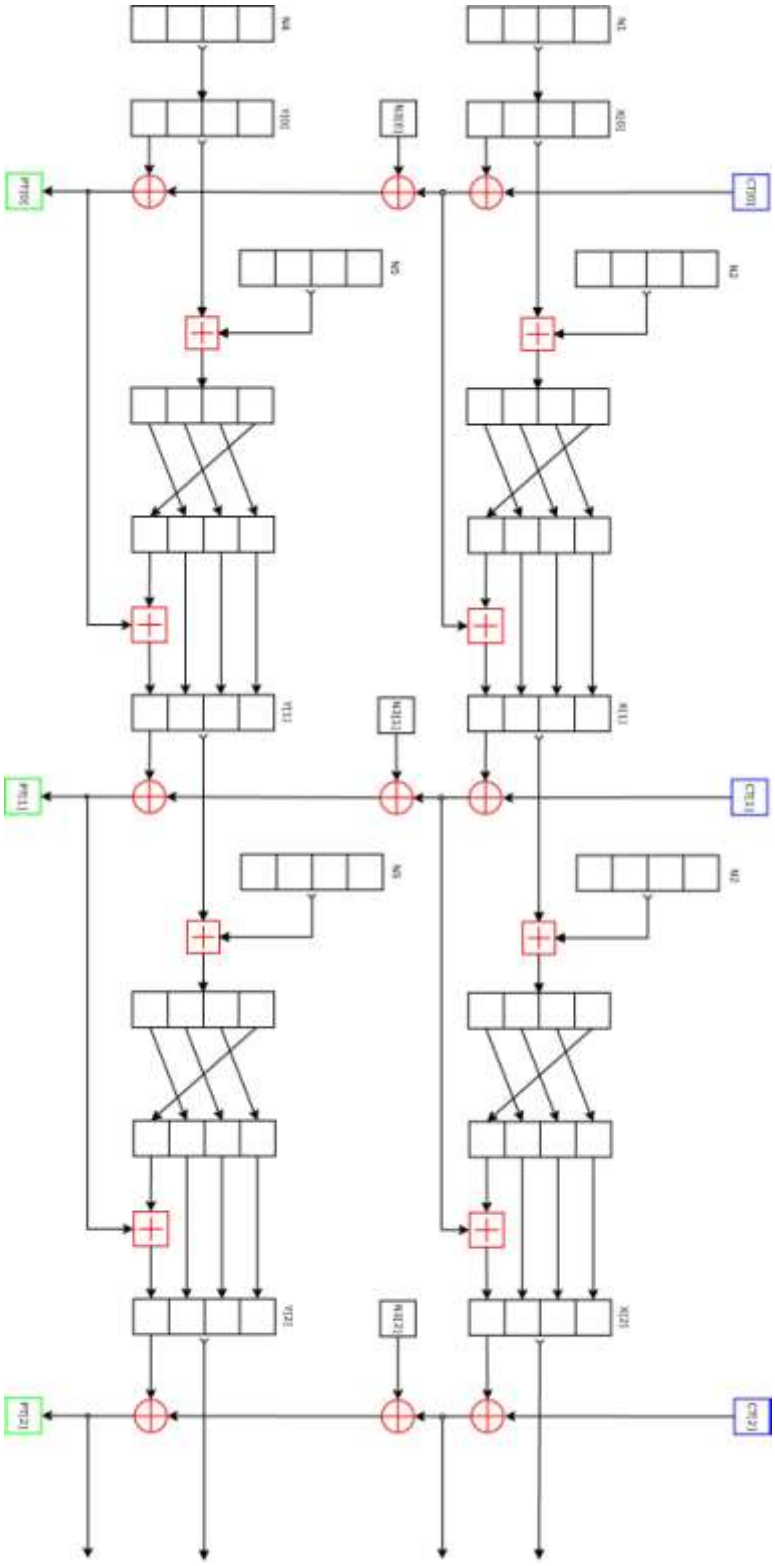
$CT[n] = PT[n] \oplus N3[n]$, $n = 0, ..., 1023$.

To by byla vzorová chyba několikanásobného použití stejného hesla. Pro luštění by pak stačilo znát jeden blok otevřeného textu a jeden blok šifrového textu, abychom vypočítali celou tabulku N3:

$N3[n] = PT[n] \oplus CT[n]$, $n = 0, ..., 1023$.

a pak odšifrovali libovolný zašifrovaný sektor.

[2] Tor je distribuovaná síť počítačů a aplikací odolná proti analýze provozu, která umožňuje anonymní přístup k síťovým serverům a anonymizaci síťových služeb.



Obr.: Schéma šifrování a dešifrování Cryptorbitem

Cryptorbit je proudová šifra, tj.
CT[n] = PT[n] ⊕ h[n], n = 0, ..., 1023,
ale do proudu hesla h[n] vkládá dvě neznámé proměnné, které jsou zpětnou vazbou ze šifrového textu[3] a zpětnou vazbou z otevřeného textu[4].

Zašifrování Cryptorbitem probíhá takto:

CT[n] = PT[n] ⊕ h[n],
h[n] = **X**[n]$_{LOW}$ ⊕ **Y**[n]$_{LOW}$ ⊕ N3[n], n = 0, ..., 1023, (H-n)

kde **X**[0], ..., **X**[1023] je zpětná vazba ze šifrového textu a **Y**[0], ..., **Y**[1023] je zpětná vazba z otevřeného textu. Přitom **X**[n] a **Y**[n] jsou 32bitové proměnné, ale při tvorbě bajtů hesla h[n] se použijí jen jejich spodní bajty, které označujeme jako **X**[n]$_{LOW}$ a **Y**[n]$_{LOW}$. Nicméně vnitřní stav zpětné vazby je 32bitový a nemůžeme ho bohužel redukovat na 8bitový. Také výpočet zpětné vazby probíhá na úrovni celých 32bitových slov.

Dešifrování Cryptorbitem je analogické:
PT[n] = CT[n] ⊕ h[n],
kde h[n] = **X**[n]$_{LOW}$ ⊕ **Y**[n]$_{LOW}$ ⊕ N3[n], n = 0, ..., 1023, (H-n)

Nyní definujeme zpětnou vazbu ze šifrového textu **X**[n]:
pro n = 0, ..., 1023:
X[n] = ((**X**[n-1] + **N2)** <<< 8) ++ (CT[n-1] ⊕ **X**[n-1]$_{LOW}$), (ZVX)
kde **X**[-1] = **N1,** CT[-1] = 0,

Doplňme pro úplnost, že v kryptologii hojně používaná operace **X** <<< 8 označuje rotaci slova X o osm bitů vlevo, tj. nejvyšší bajt se přesune na nejnižší a ostatní bajty se posunou o jednu pozici výše.

Naproti tomu **znak ++** jsme si právě vymysleli a označuje to, že **když se na 32bitové slovo X přičítá osmibitové slovo s** (jako je tomu v ZVX), tj X ++ s , tak se zanedbá aritmetický přenos (carry) z osmé pozice na devátou. Jinými slovy, **na nejnižší bajt X se načte bajt s, ale pouze v modulu 256**, tj. případné carry se nikam nepřenáší, a vyšší bajty slova X nejsou nijak ovlivněny.

Podle tohoto vzorce máme:

X[0] = ((**N1** + **N2)** <<< 8) ++ (0 ⊕ **N1**),
X[1] = ((**X**[0] + **N2)** <<< 8) ++ (CT[0] ⊕ **X**[0]),
X[2] = ((**X**[1] + **N2)** <<< 8) ++ (CT[1] ⊕ **X**[1]),

---

[3] Tato zpětná vazba závisí na předchozím šifrovém textu a navíc se do ní přimíchává i klíč N1 a N2

[4] Tato zpětná vazba závisí na předchozím otevřeném textu a navíc se do ní přimíchává i klíč N4 a N5

X[3] = ((**X**[2] + **N2)** <<< 8) ++ (CT[2] ⊕ **X**[2]),
....
X[1023] = ((**X**[1022] + **N2)** <<< 8) ++ (CT[1022] ⊕ **X**[1022]).

Nyní definujeme zpětnou vazbu z otevřeného textu **Y**[n]:
pro n = 0, ..., 1023:
Y[n] = ((**Y**[n-1] + **N5)** <<< 8) ++ PT[n-1], (ZVY)
kde **Y**[-1] **= N4,** PT[-1] = 0,

Podle tohoto vzorce máme:

Y[0] = (**N4** + **N5)** <<< 8,
Y[1] = ((**Y**[0] + **N5)** <<< 8) ++ PT[0],
Y[2] = ((**Y**[1] + **N5)** <<< 8) ++ PT[1],
Y[3] = ((**Y**[2] + **N5)** <<< 8) ++ PT[2],
....
Y[1023] = ((**Y**[1022] + **N5)** <<< 8) ++ PT[1022].

Povšimněme si důležité skutečnosti, která nás bude trápit celou dobu luštění, a to, že **jen X**[0] **a Y**[0] **je závislé pouze na klíči**, zatímco **ostatní hodnoty X[n] a Y[n]** už nějakým způsobem závisí na otevřeném nebo šifrovém textu, tj. **jsou proměnné**. To nám bude dosti ztěžovat luštění.

První bajt
Šifrování prvního bajtu otevřeného textu probíhá pochopitelně bez zpětné vazby, takže ze známého otevřeného textu bychom vypočetli první bajt hesla a tím i první informaci o klíči takto jednoduše:

h[0] = $\mathbf{X}[0]_{LOW}$ ⊕ $\mathbf{Y}[0]_{LOW}$ ⊕ N3[0] = {(**N1** + **N2)** <<< 8) ++ **N1**}$_{LOW}$ ⊕ {(**N4** + **N5)** <<< 8}$_{LOW}$ ⊕ N3[0], (H-0)

Pokud známe první bajt jednoho otevřeného sektoru, můžeme z něj h[0] prostým způsobem vypočítat jako
h[0] = CT[0] ⊕ PT[0], tj. získáme informaci o klíči:
{(**N1** + **N2)** <<< 8) ++ **N1**}$_{LOW}$ ⊕ {(**N4** + **N5)** <<< 8}$_{LOW}$ ⊕ N3[0] = CT[0] ⊕ PT[0]
neboli

$\mathbf{X}[0]_{LOW}$ ⊕ $\mathbf{Y}[0]_{LOW}$ ⊕ N3[0] = CT[0] ⊕ PT[0].

Důležité je, že **h**[0] **je stejné pro jakýkoliv sektor**, neboť zde v h[0] právě chybí zpětná vazba. To znamená, že ze znalosti otevřeného a šifrového textu **jednoho sektoru** můžeme dešifrovat **všechny první bajty u všech ostatních sektorů**:

PT[0] = CT[0] ⊕ h[0]. (PT-0)

Druhý bajt

Pro druhý bajt otevřeného a šifrového textu sektoru máme tyto vztahy:

X[0] = ((**N1** + **N2)** <<< 8) ++ (0 ⊕ **N1**),
Y[0] = (**N4** + **N5)** <<< 8,
X[1] = ((**X**[0] + **N2)** <<< 8) ++ (CT[0] ⊕ **X**[0]$_{LOW}$),
Y[1] = ((**Y**[0] + **N5)** <<< 8) ++ PT[0],
CT[1] = PT[1] ⊕ h[1],

kde h[1] = **X**[1]$_{LOW}$ ⊕ **Y**[1]$_{LOW}$ ⊕ N3[1] = {((**X**[0] + **N2)** <<< 8) ++ (CT[0] ⊕ **X**[0])}$_{LOW}$ ⊕ {((**Y**[0] + **N5)** <<< 8) ++ PT[0] }$_{LOW}$ ⊕ N3[1].

Povšimněme si, že první závorky u **X**[1] i **Y**[1] závisí pouze na klíči, nikoli na předchozím otevřeném nebo šifrovém textu, tj. jsou pro všechny sektory stejné.
Označme je
A1 = (**X**[0] + **N2)** <<< 8, (**je závislé pouze na klíči**)
B1 = (**Y**[0] + **N5)** <<< 8, (**je závislé pouze na klíči**).

Z těchto hodnot budeme při šifrování bajtu PT[1] používat jen nejnižší bajt. Označme
a1 = **A1**$_{LOW}$ = {(**X**[0] + **N2)** <<< 8}$_{LOW}$,
b1 = **B1**$_{LOW}$ = {(**Y**[0] + **N5)** <<< 8}$_{LOW}$,
a ještě
c1 = **X**[0]$_{LOW}$ ⊕ h[0] = **Y**[0]$_{LOW}$ ⊕ N3[0],
d1 = N3[1].

Potom
X[1] = **A1** ++ (CT[0] ⊕ **X**[0]$_{LOW}$).
X[1]$_{LOW}$ = (a1 + (CT[0] ⊕ **X**[0]$_{LOW}$)) mod256 = (a1 + (PT[0] ⊕ h[0] ⊕ **X**[0]$_{LOW}$)) mod256 = (a1 + (PT[0] ⊕ c1)) mod256,
Y[1] = **B1** ++ PT[0],
Y[1]$_{LOW}$ = (b1 + PT[0]) mod256,

h[1] = **X**[1]$_{LOW}$ ⊕ **Y**[1]$_{LOW}$ ⊕ N3[1] = (a1 + (PT[0] ⊕ c1)) ⊕ (b1 + PT[0]) ⊕ d1. (H1)

Nyní zkusme využívat rovnici (H1) pro různé otevřené sektory, přesněji pro sektory, které mají různé PT[1], tj. i různé h[1]. V rovnici (H1) dosadíme za h[1] a dostáváme

CT[1] ⊕ PT[1] = (a1 + (PT[0] ⊕ c1)) ⊕ (b1 + PT[0]) ⊕ d1, (abcd-1)

kde bajty a1, b1, c1, d1 jsou neznámé bajty, **závislé pouze na pevném klíči**, zatímco **bajty PT**[0]**, PT**[1] **a CT**[1] **jsou známé a mění se**.

Pro různé sektory, kde známe PT[0] i PT[1], tak dostáváme vždy jednu rovnici (abcd-1) pro neznámé bajty a1, b1, c1, d1. Řekněme pro jednoduchost, že ze soustavy například osmi těchto rovnic (pro čtyři neznámé by osm rovnic mohlo dostačovat, ale ve skutečnosti je to složitější otázka) bychom dokázali **určit čtyři neznámé bajty a1 až d1**.

Potom pro sektor, kde neznáme PT[0], ani PT[1] můžeme určit jak PT[0], tak PT[1]:

A. PT[0] = CT[0] ⊕ h[0], kde h[0] je známé a stejné pro všechny sektory (z minulého kroku) a ze znalosti PT[0] můžeme dopočítat
B. PT[1] = CT[1] ⊕ (a1 + (PT[0] ⊕ c1)) ⊕ (b1 + PT[0]) ⊕ d1, kde a, b, c, d jsou známé a stejné pro všechny sektory

Jinými slovy, **pokud známe bajt PT**[0] **pro jeden sektor a bajt PT**[1] **pro osm sektorů, umíme dešifrovat bajty PT**[0] **i PT**[1] **pro jakýkoliv sektor**.

Poznámka:
Hodnoty a1, b1, c1, d1 jsou skutečně stejné pro všechny sektory, neboť závisí pouze na klíči:
Máme
X[0] = (**N1** + **N2**) <<< 8 ++ **N1**,
Y[0] = (**N4** + **N5**) <<< 8,
tedy **X**[0] i **Y**[0] jsou závislé pouze na klíči, a
a1 = {(**X**[0] + **N2**) <<< 8}$_{LOW}$
b1 = {(**Y**[0] + **N5**) <<< 8}$_{LOW}$
c1 = **Y**[0]$_{LOW}$ ⊕ N3[0]
d1 = N3[1],
jsou tedy také závislé pouze na klíči.

Shrneme nyní, co víme:
Při analýze prvního bajtu jsme byli schopni určit hodnotu h[0] = **X**[0]$_{LOW}$ ⊕ **Y**[0]$_{LOW}$ ⊕ N3[0], závislou pouze na klíči.
Při analýze druhého bajtu jsme určili hodnotu c1 = **Y**[0]$_{LOW}$ ⊕ N3[0], tj. nyní známe i hodnotu **X**[0]$_{LOW}$. Dále jsme určili a1, b1 a d1, PT[0], PT[1], h[0], h[1].
Máme **X**[1]$_{LOW}$ = (a1 + (CT[0] ⊕ **X**[0]$_{LOW}$)) mod256, přičemž všechny hodnoty na pravé straně již známe, tedy známe i **X**[1]$_{LOW}$.
Dále máme **Y**[1]$_{LOW}$ = (b1 + PT[0]) mod256, přičemž všechny hodnoty na pravé straně již známe, tedy známe i **Y**[1]$_{LOW}$.

Poznámka:
X[0] = (**N1** + **N2**) <<< 8 ++ **N1**, známe **X**[0]$_{LOW}$
Y[0] = (**N4** + **N5**) <<< 8, známe pouze **Y**[0]$_{LOW}$ ⊕ N3[0],
X[1] = ((**X**[0] + **N2**) <<< 8) ++ (CT[0] ⊕ **X**[0]$_{LOW}$), známe **X**[1]$_{LOW}$
Y[1] = ((**Y**[0] + **N5**) <<< 8) ++ PT[0], známe **Y**[1]$_{LOW}$
A1 = (**X**[0] + **N2**) <<< 8, závislé pouze na klíči, známe a1 = {(**X**[0] + **N2**) <<< 8}$_{LOW}$
B1 = (**Y**[0] + **N5**) <<< 8, závislé pouze na klíči, známe b1 = {(**Y**[0] + **N5**) <<< 8}$_{LOW}$
X[1] = **A1** ++ (CT[0] ⊕ **X**[0]$_{LOW}$),
Y[1] = **B1** ++ PT[0],
známe N3[1], h[0].

Třetí bajt
Pro 3. bajt otevřeného a šifrového textu sektoru máme tyto vztahy:
X[0] = ((**N1** + **N2**) <<< 8) ++ (0 ⊕ **N1**),
Y[0] = (**N4** + **N5**) <<< 8,
A1 = (**X**[0] + **N2**) <<< 8, závislé pouze na klíči, známe pouze a1 = {(**X**[0] + **N2**) <<< 8}$_{LOW}$
B1 = (**Y**[0] + **N5**) <<< 8, závislé pouze na klíči, známe pouze b1 = {(**Y**[0] + **N5**) <<< 8}$_{LOW}$

$\mathbf{X}[1] = \mathbf{A1}$ ++ $(CT[0] \oplus \mathbf{X}[0]_{LOW})$,
$\mathbf{Y}[1] = \mathbf{B1}$ ++ $PT[0]$,
$\mathbf{X}[2] = ((\mathbf{X}[1] + \mathbf{N2}) <<< 8)$ ++ $(CT[1] \oplus \mathbf{X}[1]_{LOW})$
$\mathbf{Y}[2] = ((\mathbf{Y}[1] + \mathbf{N5}) <<< 8)$ ++ $PT[1]$
$CT[2] = PT[2] \oplus h[2]$,
$h[2] = \mathbf{X}[2]_{LOW} \oplus \mathbf{Y}[2]_{LOW} \oplus N3[2]$

Vypočítáme opět zpětnou vazbu z otevřeného textu a ze šifrového textu a pokusíme se je vyjádřit odděleně pomocí hodnot závislých pouze na klíči a hodnot proměnných. Bohužel se nám tam začnou plést bity přenosu mezi jednotlivými bajty při scítání 32bitových slov. Proto také musíme jednotlivé bajty 32bitového slova označit. Obecně tedy značíme pro 32bitové slovo **W** jeho bajty od nejvyššího k nejnižšímu: $\mathbf{W}_{3RD}$, $\mathbf{W}_{2ND}$, $\mathbf{W}_{1ST}$, $\mathbf{W}_{LOW}$.

Máme
$\mathbf{X}[2]_{LOW} = ((\mathbf{X}[1] + \mathbf{N2}) <<< 8)_{LOW}$ ++ $(CT[1] \oplus \mathbf{X}[1]_{LOW}) =$
$= (\mathbf{X}[1] + \mathbf{N2})_{3RD} + (CT[1] \oplus \mathbf{X}[1]_{LOW}) =$
$= (\{\mathbf{A1}$ ++ $(CT[0] \oplus \mathbf{X}[0]_{LOW}\ )\} + \mathbf{N2})_{3RD} + (CT[1] \oplus \{ a1 + (CT[0] \oplus \mathbf{X}[0]_{LOW}\ )\})$ =
$= \mathbf{A1}_{3RD} + \mathbf{N2}_{3RD} + a2carry + (CT[1] \oplus \{ a1 + (CT[0] \oplus \mathbf{X}[0]_{LOW}\ )\})$
kde a2carry je bit přenosu z 2.bajtu na 3.bajt (číslování bajtů od nuly) při sčítání $\{\mathbf{A1}$ ++ $(CT[0] \oplus \mathbf{X}[0]_{LOW}\ )\} + \mathbf{N2}$,

$\mathbf{Y}[2]_{LOW} = \{((\mathbf{Y}[1] + \mathbf{N5}) <<< 8)$ ++ $PT[1]\ \}_{LOW} =$
$= \{((\mathbf{[}\ \mathbf{B1}$ ++ $PT[0]\ \mathbf{]} + \mathbf{N5}) <<< 8)$ ++ $PT[1]\ \}_{LOW} =$
$= (\mathbf{[}\ \mathbf{B1}$ ++ $PT[0]\ \mathbf{]} + \mathbf{N5})_{3RD} + PT[1] =$
$= \mathbf{B1}_{3RD} + \mathbf{N5}_{3RD} + b2carry + PT[1]$,
kde b2carry je bit přenosu z 2.bajtu na 3.bajt (číslování bajtů od nuly) při sčítání $(\mathbf{[}\ \mathbf{B1}$ ++ $PT[0]\ \mathbf{]} + \mathbf{N5}$,

$h[2] = CT[2] \oplus PT[2] = \mathbf{X}[2]_{LOW} \oplus \mathbf{Y}[2]_{LOW} \oplus N3[2] =$
$= \{\ \mathbf{A1}_{3RD} + \mathbf{N2}_{3RD} + a2carry + (CT[1] \oplus \{ a1 + (CT[0] \oplus \mathbf{X}[0]_{LOW}\ )\})\ \}$
$\oplus \{\ \mathbf{B1}_{3RD} + \mathbf{N5}_{3RD} + b2carry + PT[1]\ \} \oplus N3[2] =$

Nyní pro přesnost a přehlednost u a2carry napíšeme index CT[0], což bude vyjadřovat závislost na CT[0], a podobně u b2carry budeme psát index PT[0]. Označme ještě $t2 = \mathbf{A1}_{3RD} + \mathbf{N2}_{3RD}$, $u2 = \mathbf{B1}_{3RD} + \mathbf{N5}_{3RD}$. **<u>Důležité je, že t2 a u2 jsou už hodnoty závislé pouze na klíči.</u>**

Předchozí rovnice má nyní tvar
$CT[2] \oplus PT[2] =$
$= \mathbf{X}[2]_{LOW} \oplus \mathbf{Y}[2]_{LOW} \oplus N3[2] =$
$= \{\ t2 + a2carry_{CT[0]} + (CT[1] \oplus \{ a1 + (CT[0] \oplus \mathbf{X}[0]_{LOW}\ )\})\} \oplus$
$\oplus \{\ u2 + b2carry_{PT[0]} + PT[1]\ \} \oplus N3[2]$, neboli

$N3[2] = CT[2] \oplus PT[2] \oplus \{\ t2 + a2carry_{CT[0]} + (CT[1] \oplus \{ a1 + (CT[0] \oplus \mathbf{X}[0]_{LOW}\ )\})\} \oplus$
$\oplus \{\ u2 + b2carry_{PT[0]} + PT[1]\ \}$, (abcd-2carry)

Opět uvažujeme osm těchto rovnic, kde vystupují různé známé hodnoty CT[2], CT[1], CT[0], PT[2], PT[1], PT[0] a neznámé bajty N3[2], u2, t2 a osm dvojic neznámých bitů $a2carry_{CT[0]}$ a $b2carry_{PT[0]}$. Připomínáme, že tyto bity jsou v každé z osmi rovnic obecně jiné, neboť závisí na konkrétních (jiných) hodnotách CT[0] a PT[0] v každé z osmi rovnic.

Řešení této soustavy je možné hrubou silou, kdy zkoušíme všechny možné kombinace hodnot t2 a a2. Z každé z osmi rovnic obdržíme čtyři možná řešení N3[2], neboť jsou čtyři kombinace bitů (a2carry, b2carry). Řešení se ale musí objevovat v průniku všech osmi množin čtyř hodnot, takže je pravděpodobné, že průnikem všech řešení všech osmi rovnic bude řešení jediné. Důležité je, že **získáme hodnoty N3[2], u2, t2**, přičemž osm dvojic hodnot $a2carry_{CT[0]}$ a $b2carry_{PT[0]}$ zatím nebudeme využívat.

Pomocí známých hodnot N3[2], u2, t2 pak **dešifrujeme třetí bajt otevřeného textu u každého sektoru**, který neznáme, avšak máme bohužel (díky neznámým bitům $a2carry_{CT[0]}$ a $b2carry_{PT[0]}$ pro tento konkrétní sektor) až čtyři možné hodnoty pro tento bajt:

$$PT[2] = CT[2] \oplus N3[2] \oplus \{\ t2 + a2carry_{CT[0]} + (CT[1] \oplus \{\ a1 + (CT[0] \oplus \mathbf{X}[0]_{LOW}\ )\})\} \oplus$$
$$\oplus \{\ u2 + b2carry_{PT[0]} + PT[1]\ \}.$$

Poznámka – celkový stav:
$\mathbf{X}[0] = (\mathbf{N1} + \mathbf{N2}) <<< 8 \ ++ \mathbf{N1}$, známe $\mathbf{X}[0]_{LOW}$
$\mathbf{Y}[0] = (\mathbf{N4} + \mathbf{N5}) <<< 8$, známe $\mathbf{Y}[0]_{LOW} \oplus N3[0]$,
$\mathbf{X}[1] = ((\mathbf{X}[0] + \mathbf{N2}) <<< 8) \ ++ (CT[0] \oplus \mathbf{X}[0]_{LOW})$, známe $\mathbf{X}[1]_{LOW}$
$\mathbf{Y}[1] = ((\mathbf{Y}[0] + \mathbf{N5}) <<< 8) \ ++ PT[0]$, známe $\mathbf{Y}[1]_{LOW}$
$\mathbf{X}[2] = ((\mathbf{X}[1] + \mathbf{N2}) <<< 8) \ ++ (CT[1] \oplus \mathbf{X}[1]_{LOW})$, až na a2carry známe $\mathbf{X}[2]_{LOW}$
$\mathbf{Y}[2] = ((\mathbf{Y}[1] + \mathbf{N5}) <<< 8) \ ++ PT[1]$, až na b2carry známe $\mathbf{Y}[2]_{LOW}$
známe d1 = N3[1], N3[2], h[0].

$\mathbf{X}[1] = \mathbf{A1} \ ++ (CT[0] \oplus \mathbf{X}[0]_{LOW})$,
$\mathbf{Y}[1] = \mathbf{B1} \ ++ PT[0]$,
$\mathbf{X}[2]_{LOW} = t2 + a2carry + (CT[1] \oplus \{\ a1 + (CT[0] \oplus \mathbf{X}[0]_{LOW}\ )\})$
$\mathbf{Y}[2]_{LOW} = \mathbf{B1}_{3RD} + \mathbf{N5}_{3RD} + b2carry + PT[1]$,

Čtvrtý bajt
Pro 4. bajt otevřeného a šifrového textu sektoru máme tyto vztahy:
$\mathbf{X}[3] = ((\mathbf{X}[2] + \mathbf{N2}) <<< 8) \ ++ (CT[2] \oplus \mathbf{X}[2]_{LOW})$
$\mathbf{Y}[3] = ((\mathbf{Y}[2] + \mathbf{N5}) <<< 8) \ ++ PT[2]$

$CT[3] = PT[3] \oplus h[3]$,
$h[3] = \mathbf{X}[3]_{LOW} \oplus \mathbf{Y}[3]_{LOW} \oplus N3[3]$

Předpokládáme, že nejpozději tady to čtenáře přestane bavit. A není divu, další práce je úmorné a nezáživné honění bitů z kouta do kouta. A běda, když uděláme chybu. Nicméně to dokončeme, i když čtenáři mohou poposkočit na konec :-)

$\mathbf{X}[3]_{LOW}$ = $((\mathbf{X}[2] + \mathbf{N2}) <<< 8)_{LOW}$ ++ $(CT[2] \oplus \mathbf{X}[2]_{LOW})$.
Připomínáme, že z třetího kroku máme
$\mathbf{X}[2]_{LOW}$ = t2 + $a2carry_{CT[0]}$ + $(CT[1] \oplus \{ a1 + (CT[0] \oplus \mathbf{X}[0]_{LOW} \mathbf{)\}})$, přičemž $a2carry_{CT[0]}$ můžeme předpokládat, že známe, protože k luštění třetího bajtu jsme se dostali přes druhý bajt.

Dále upravíme výraz
$(\mathbf{X}[2] + \mathbf{N2}) <<< 8)_{LOW} = (\mathbf{X}[2] + \mathbf{N2})_{3RD} = \mathbf{X}[2]_{3RD} + \mathbf{N2}_{3RD}$ + c3carry.
dále $\mathbf{X}[2]_{3RD} = (((\mathbf{X}[1] + \mathbf{N2}) <<< 8)$ ++ $(CT[1] \oplus \mathbf{X}[1]_{LOW}))_{3RD}$ =
= $((\mathbf{X}[1] + \mathbf{N2}) <<< 8)_{3RD} = (\mathbf{X}[1] + \mathbf{N2})_{2ND}$ =
= $(\{\mathbf{A1}$ ++ $(CT[0] \oplus \mathbf{X}[0]_{LOW})\} + \mathbf{N2})_{2ND} = \mathbf{A1}_{2ND} + \mathbf{N2}_{2ND}$ + d3carry

kde
d3carry je bit přenosu z 1.bajtu na 2.bajt při sčítání $\mathbf{X}[1] + \mathbf{N2}$,
c3carry je bit přenosu z 2.bajtu na 3.bajt při sčítání $\mathbf{X}[2] + \mathbf{N2}$,

Celkově tedy dosadíme do výrazu pro $\mathbf{X}[3]_{LOW}$
= $(\mathbf{X}[2]_{3RD} + \mathbf{N2}_{3RD}$ + c3carry) ++ $(CT[2] \oplus \mathbf{X}[2]_{LOW})$ =
= $(\mathbf{A1}_{2ND} + \mathbf{N2}_{2ND}$ + d3carry + $\mathbf{N2}_{3RD}$ + c3carry) + $(CT[2] \oplus \mathbf{X}[2]_{LOW})$ =
= $(\mathbf{A1}_{2ND} + \mathbf{N2}_{2ND}$ + d3carry + $\mathbf{N2}_{3RD}$ + c3carry) +
+ $(CT[2] \oplus \{$ t2 + $a2carry_{CT[0]}$ + $(CT[1] \oplus \{ a1 + (CT[0] \oplus \mathbf{X}[0]_{LOW} \mathbf{)\}}) \}$).

Nyní budeme počítat $\mathbf{Y}[3]_{LOW}$

$\mathbf{Y}[3]_{LOW}$ = $\{((\mathbf{Y}[2] + \mathbf{N5}) <<< 8)$ ++ $PT[2] \}_{LOW}$ =
= $((\mathbf{Y}[2] + \mathbf{N5}) <<< 8)_{LOW} + PT[2]_{LOW}$ =
= $(\mathbf{Y}[2] + \mathbf{N5})_{3RD} + PT[2]_{LOW}$ =
= $\mathbf{Y}[2]_{3RD} + \mathbf{N5}_{3RD}$ + b3carry + $PT[2]_{LOW}$ =
= $\mathbf{[}((\mathbf{Y}[1] + \mathbf{N5}) <<< 8)$ ++ $PT[1]\mathbf{]}_{3RD} + \mathbf{N5}_{3RD}$ + b3carry + $PT[2]_{LOW}$ =
= $(\mathbf{Y}[1] + \mathbf{N5})_{2ND} + \mathbf{N5}_{3RD}$ + b3carry + $PT[2]_{LOW}$
= $\mathbf{Y}[1]_{2ND} + \mathbf{N5}_{2ND}$ + a3carry + $\mathbf{N5}_{3RD}$ + b3carry + $PT[2]_{LOW}$
= $\mathbf{[B1}$ ++ $PT[0]\mathbf{]}_{2ND} + \mathbf{N5}_{2ND}$ + a3carry + $\mathbf{N5}_{3RD}$ + b3carry + $PT[2]_{LOW}$
= $\mathbf{B1}_{2ND} + \mathbf{N5}_{2ND}$ + a3carry + $\mathbf{N5}_{3RD}$ + b3carry + $PT[2]_{LOW}$
kde b3carry je bit přenosu z 2.bajtu na 3.bajt při sčítání $\mathbf{Y}[2] + \mathbf{N5}$,
kde a3carry je bit přenosu z 1.bajtu na 2.bajt při sčítání $\mathbf{Y}[1] + \mathbf{N5}$,

Nyní dosadíme $\mathbf{X}[3]_{LOW}$ a $\mathbf{Y}[3]_{LOW}$ do rovnice pro h[3]:

h[3] = $\mathbf{X}[3]_{LOW} \oplus \mathbf{Y}[3]_{LOW} \oplus$ N3[3] =
{
$(\mathbf{A1}_{2ND} + \mathbf{N2}_{2ND}$ + d3carry + $\mathbf{N2}_{3RD}$ + c3carry) +
+ $\big($CT[2] $\oplus \{$ t2 + $a2carry_{CT[0]}$ + $(CT[1] \oplus \{ a1 + (CT[0] \oplus \mathbf{X}[0]_{LOW} \mathbf{)\}}) \}\big)$
}$\oplus$
{
$\mathbf{B1}_{2ND} + \mathbf{N5}_{2ND}$ + a3carry + $\mathbf{N5}_{3RD}$ + b3carry + $PT[2]_{LOW}$
}$\oplus$ N3[3], takže máme

$CT[3] \oplus PT[3] =$
$=$
$\{$
$(\mathbf{A1}_{2ND} + \mathbf{N2}_{2ND} + d3carry + \mathbf{N2}_{3RD} + c3carry) +$
$+ \big(CT[2] \oplus \{ t2 + a2carry_{CT[0]} + (CT[1] \oplus \{ a1 + (CT[0] \oplus \mathbf{X}[0]_{LOW}) \}) \}\big)$
$\}\oplus$
$\{$
$\mathbf{B1}_{2ND} + \mathbf{N5}_{2ND} + a3carry + \mathbf{N5}_{3RD} + b3carry + PT[2]_{LOW}$
$\}\oplus N3[3],$

Když označíme
$v3 = \mathbf{A1}_{2ND} + \mathbf{N2}_{2ND} + \mathbf{N2}_{3RD}$ a
$w3 = \mathbf{B1}_{2ND} + \mathbf{N5}_{2ND} + \mathbf{N5}_{3RD}$, pak dostáváme

$N3[3] =$
$= CT[3] \oplus PT[3] \oplus$
$\{ (v3 + d3carry_{CT[0]} + c3carry_{CT[0,1]}) +$
$+ \big(CT[2] \oplus \{ t2 + a2carry_{CT[0]} + (CT[1] \oplus \{ a1 + (CT[0] \oplus \mathbf{X}[0]_{LOW}) \}) \}\big)$
$\}\oplus \{w3 + a3carry_{PT[0]} + b3carry_{PT[1,2]} + PT[2]_{LOW} \}$ (abcd-3carry)

<u>Důležité je, že v3 a w3 jsou hodnoty závislé pouze na klíči.</u>

Opět uvažujeme osm těchto rovnic, kde vystupují různé známé hodnoty CT[3], CT[2], CT[1], CT[0], PT[3], PT[2], PT[1], PT[0], $a2carry_{CT[0]}$, $b3carry_{PT[1]}$ a neznámé bajty N3[3], v3, w3 a 8 čtveřic neznámých bitů d3carry, c3carry, b3carry a a3carry. Připomínáme, že tyto bity jsou v každé z osmi rovnic obecně jiné, neboť závisí na konkrétních (jiných) hodnotách CT[0,1,2] a PT[0,1,2] v každé z osmi rovnic.

Řešení této soustavy je možné hrubou silou, kdy zkoušíme všechny možné kombinace hodnot v3 a w3 a N3[3], přičemž z každé rovnice obdržíme 16 hodnot N3[3], neboť je 16 kombinací bitů (a3carry, b3carry, c3carry, d3carry). Řešení se musí objevovat v průniku všech osmi množin, takže je pravděpodobné, že průnikem všech řešení všech osmi rovnic bude řešení jediné. Důležité je, že získáme hodnoty **<u>N3[3], v3, w3</u>**, přičemž 8 čtveřic hodnot (a3carry, b3carry, c3carry, d3carry) zatím nebudeme využívat.

Pomocí známých hodnot N3[3], v3, w3 pak můžeme s využitím (abcd-3carry) **<u>dešifrovat čtvrtý bajt otevřeného textu u každého sektoru</u>**, avšak máme bohužel (díky neznámým bitům carrypro tento konkrétní sektor) až 16 možných hodnot pro tento bajt.

Poznámka – celkový stav:
$\mathbf{X}[0] = (\mathbf{N1} + \mathbf{N2}) <<< 8 \; ++ \mathbf{N1}$, známe $\mathbf{X}[0]_{LOW}$
$\mathbf{Y}[0] = (\mathbf{N4} + \mathbf{N5}) <<< 8$, známe $\mathbf{Y}[0]_{LOW} \oplus N3[0]$,
$\mathbf{X}[1] = ((\mathbf{X}[0] + \mathbf{N2}) <<< 8) \; ++ (CT[0] \oplus \mathbf{X}[0]_{LOW})$, známe $\mathbf{X}[1]_{LOW}$
$\mathbf{Y}[1] = ((\mathbf{Y}[0] + \mathbf{N5}) <<< 8) \; ++ PT[0]$, známe $\mathbf{Y}[1]_{LOW}$
$\mathbf{X}[2] = ((\mathbf{X}[1] + \mathbf{N2}) <<< 8) \; ++ (CT[1] \oplus \mathbf{X}[1]_{LOW})$, až na a2carry známe $\mathbf{X}[2]_{LOW}$
$\mathbf{Y}[2] = ((\mathbf{Y}[1] + \mathbf{N5}) <<< 8) \; ++ PT[1]$, až na b2carry známe $\mathbf{Y}[2]_{LOW}$

X[3] = ((**X**[2] + **N2)** <<< 8) ++ (CT[2] ⊕ **X**[2]$_{LOW}$), známe v3 = (**X**[1] + **N2)**$_{2ND}$ + **N2**$_{3RD}$
Y[3] = ((**Y**[2] + **N5)** <<< 8) ++ PT[2], známe w3 = (**Y**[1] + **N5)**$_{2ND}$ + **N5**$_{3RD}$,
známe d1 = N3[1], N3[2], N3[3], h[0].

neznáme: **X**[0], **Y**[0], **N2**, **N5**, což je 16 neznámých bajtů,
známe:
X[0]$_{LOW}$
Y[1]$_{LOW}$
X[1]$_{LOW}$
(**X**[1] + **N2**)$_{3RD}$,
(**Y**[1] + **N5**)$_{3RD}$,
(**X**[1] + **N2)**$_{2ND}$ + **N2**$_{3RD}$,
(**Y**[1] + **N5)**$_{2ND}$ + **N5**$_{3RD}$,
což je 8 bajtů.

Zbývá nám tedy určit minimálně ještě 8 bajtů, což tímto tempem bude trvat 4-5 kroků, podobných jako předchozí. ..

Tady tento článek končí, protože jsme zatím více ve svém volnu nestihli. Rádi bychom čtenáře požádali, aby se zamysleli a zkusili přijít na elegantnější způsob luštění, protože data má tímto způsobem zašifrovaný jeden obecní úřad, který nemá prostředky na to, abychom jim mohli najmout několik lidí, vysvětlit jim podstatu a oni by to mohli během třeba měsíce dovést do konečného doluštění a sepsání programu, který data vrátí zpět do podoby před řáděním Cryptorbitu. Zdá se, že věc je poměrně myšlenkově jednoduchá, ale co když nám bity carry v dalších krocích přeroustou přes hlavu? Dojdeme vůbec k řešení? Nechtělo by to někoho s jiným nápadem na elegantnější řešení?

Ještě experimentální výsledky. Na předmětném počítači se nám podařilo zatím nalézt 26 zašifrovaných souborů, ke kterým známe PT. Jsou to například soubory s ukázkami obrázků a hudby (Lekníny.jpg, Modré vrcholky.jpg, Beethovenova symfonie č. 9 (Scherzo).wma,...), které jsou na všech instalacích Windows stejné. Vir se důsledně vyhýbá souborům .EXE, .DLL a podobně, nelze tedy pro luštění použít známý PT od aplikací. Stejně tak ignoruje adresáře Windows a Program files. Nalezli jsme ale například soubory s jazykovou lokalizací pro Total Commander (WCMD_CHN.LNG, WCMD_FRA.LNG,...).

Pro 19 dvojic PT-OT mají rovnice pro 2. byte 16 řešení, rovnice pro 3. byte mají 128 řešení a pro 4. byte je 1024 řešení včetně všech kombinací carry. Pro více souborů z našeho vzorku se počet řešení zatím nesnižuje.

C. Pár poznámek k šifrátoru MAFFIE

Pavel Vondruška, pavel.vondruska@crypto-world.info

Většina lidí si pod pojmem speciální šifrovací zařízení vybaví v prvé řadě německý šifrovací stroj Enigma. Pokud se dále zeptáte, kdy se začala používat tato šifrovací zařízení, tak většina pravděpodobně uvede období mezi dvěma světovými válkami. Z literatury je totiž dostatečně známo, že během 1. světové války se žádné kryptografické zařízení neproslavilo a toto období je charakteristické používáním kódových knih a různě silných a složitých ručních šifer.

Řadě studentů informační bezpečnosti / kryptografie je však známo, že se návrhy kryptografických zařízení objevily podstatně dříve. Zpravidla se v této souvislosti připomíná velmi zajímavý šifrovací stroj Thomase Jeffersona z roku 1795 (!), který se skládal z 36 pohyblivých disků navlečených na společné ose. Ovšem současně je známo, že zařízení, které na shodném principu nezávisle sestrojil důstojník Etienne Bazeries, se prakticky využívalo opět až v meziválečném období. Konkrétně jej používala americká armáda od roku 1923 do roku 1942 a je známé jako M-94.

Jak jsme na tom u nás (tj. v ČR / SR)?

Ve své prezentaci *Vývoj kryptografických zařízení v ČS(S)R* na MKB 2009 jsem začal popis výroby šifrátorů také až rokem 1930. Konkrétně jsem zahájil popisem výroby a použití zařízení ŠTOLBA.

Historie vývoje československých šifrátorů začala ve třicátých letech minulého století. První známý šifrátor československé výroby byl navržen pravděpodobně okolo roku 1930 odborníkem na mechanické konstrukce plk. Ing. Štolbou. Šifrátor proto dostal označení ŠTOLBA. Vyrobeno bylo asi 50 až 55 kusů, které byly pravděpodobně používány na ministerstvech a u vedení Československé armády. Jeden šifrátor byl údajně přidělen k osobnímu používání prezidentu Edvardu Benešovi. Do roku 1938 byl jediným šifrovacím strojem, který byl vyvinut v Československu a navíc ještě vyráběn ve větší sérii. Šifrovací stroj ŠTOLBA představoval svou konstrukcí zcela ojedinělé řešení mechanického šifrátoru s vlastní tvorbou hesla. Princip šifrování textu byl sice analogický způsobu šifrování u německého šifrovacího stroje Enigma, ale přenos „signálu“ z klávesnice přes šifrová kola až po tiskový mechanismus byl řešen zcela specifickým způsobem a to pneumatickým převodem. O využití tohoto stroje během 2. světové války není nic známo. V letech 1945 – 1955 byl používán v československé armádě k výrobě náhodného hesla, které bylo pak využíváno v jiných šifrových systémech. [3]

Zcela určitě to však není první šifrovací stroj, který byl u nás používán. Ve své knize Kryptologie, šifrování a tajná písma [5] jsem u hesla ***1914-1918*** uvedl známé svědectví o existenci šifrátoru, který byl použit v domácím odboji.

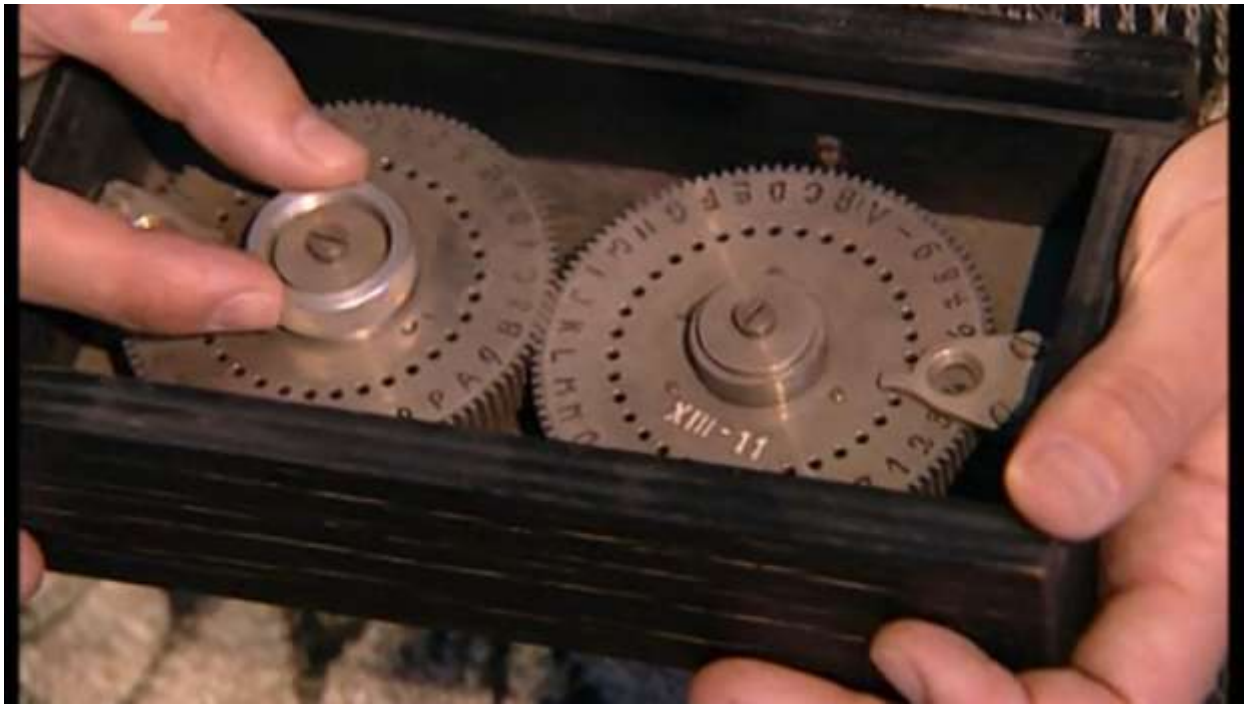
1914-1918
Tajná komunikace české protirakouské politické opozice během první světové války je dobře vylíčena v knize Karla Čapka Hovory s T. G. Masarykem.
Prezident Tomáš Garrigue Masaryk v ní vzpomíná:
...
Pro psaní senzitivních zpráv mezi domácím a zahraničním odbojem se používal i neviditelný inkoust, který byl za 1.světové války velmi rozšířen a s oblibou používán špiony na všech frontách. Konkrétně to bylo v korespondenci mezi pražským průmyslníkem J. J. Fričem a jeho obchodním zástupcem v Itálii a Švýcarsku Z. Rohlou.

Mimo těchto typických steganografických metod se používalo i šifrování. Svědectví o jeho použití se dochovalo v již zmíněné knize, kde prezident Masaryk uvádí, že inženýr Baráček v Ženevě pro účely odboje ***zkonstruoval dokonce šifrovací stroj****.*

O tomto šifrátoru jsem se při popisu vývoje šifrátorů u nás dosud nezmiňoval, neboť jsem se domníval, že mimo uvedené zmínky se k němu již nedá získat žádný konkrétní materiál a šifrátor je „ztracen“. Ovšem to jsem se velmi mýlil.

Nedávno jsem dostal od kolegy RNDr. Jozefa Krajčoviče následující velmi zajímavý e-mail [4]

…, *nasiel som v knihe Kryptograficky front velkej vlasteneckej, L. S. Butyrskeho, D. A. Larina a G. P. Sankina, zaujimavu chybu. Popisuju v nej na s. 63-64 sifrovaci stroj, znazorneny na obrazku, pochadzajucom z expozicie muzea VHU v Prahe a ktory spomina aj T. G. Masaryk vo svojich rozhovoroch s Karlom Capkom, zrejme navrhnuty a zostrojeny ing. Barackom.*



Nepochopenie vtedajsich realii ich zaviedlo na scestie a pisu tam o nom ako o ***prostriedku pre utajovanie sprav "zlocineckych skupin a kontrabandistov",*** *tj.* ***mafie a nie Maffie*** *ako o zakonspirovanej odbojovej organizacii, ktorej clenmi boli ako je zname Edvard Beneš, Karel Kramar, Alois Rasin, Josef Scheiner a Premysl Samal. Jej ulohou bolo udrzovat styk s Masarykom aj prostrednictvom sifrovanej korespondencie..* [6]

Ano, mafie je obecně vnímána jako tajné zájmové sdružení používající nelegální (zločinné) metody, zpravidla se spojuje především s „původem“ tajné společnosti vzniklé v polovině 19. století na Sicílii známé jako Cosa Nostra („naše věc“).

Ne každý (zejména mladší) si vybaví „naši“ Maffii, která vznikla a působila na začátku 20. století v Rakousku - Uhersku a svými cíli a metodami se zcela zásadně liší od výše uvedené „mafie“. Tato Maffie byl hlavní orgán českého domácího odboje během první světové války. Řídila zpravodajskou a konspirační činnost, předávání informací a udržovala spojení se zahraniční sekcí. Její činnosti se účastnilo přes 200 osob.

Pro doplnění si dále uvedeme, co k tomu uvádí Wikipedie [7]:

Po odjezdu profesora T. G. Masaryka do zahraničí v prosinci 1914 byl utvořen domácí výbor nazvaný po vzoru sicilské mafie "maffie". Od začátku podporovala zahraniční směřování akcí profesora T. G. Masaryka a stála na důsledně protirakouských a prodohodových pozicích. V březnu 1915 se ustavilo její předsednictvo: Edvard Beneš, Karel Kramář, Alois Rašín, Josef Scheiner a Přemysl Šámal.

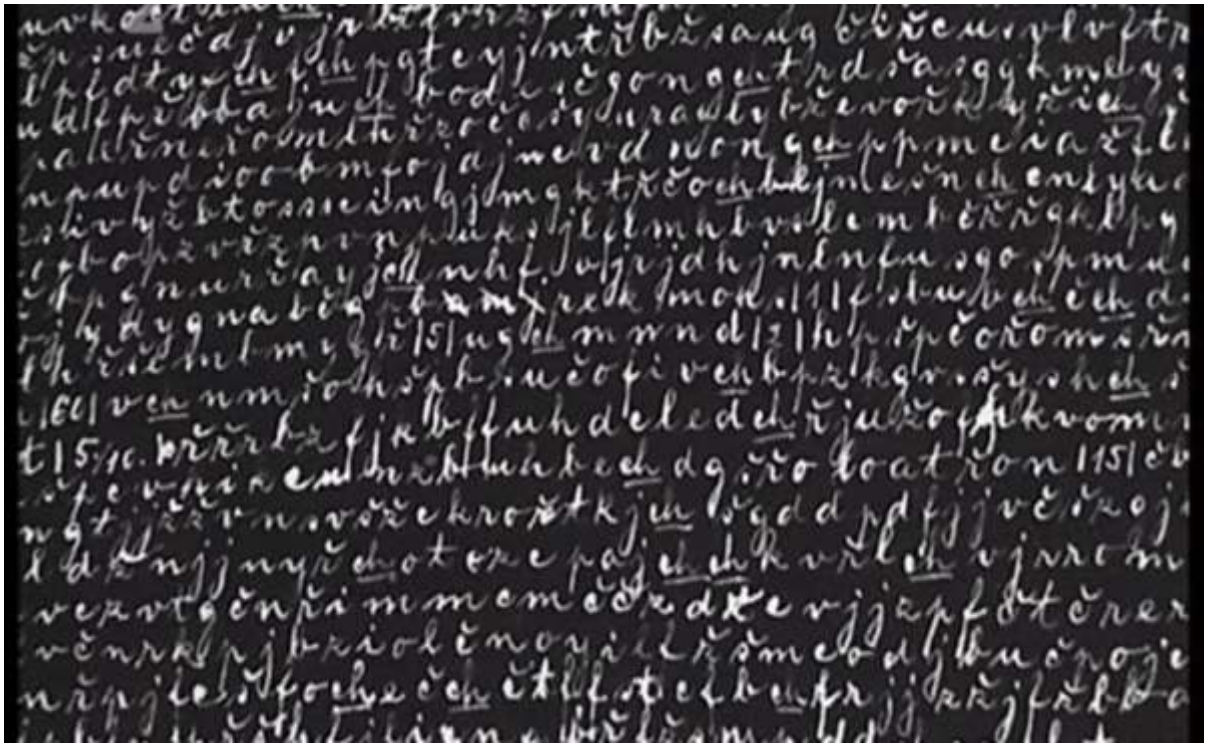
V září 1915 se po nucené emigraci E. Beneše stal jejím vedoucím Přemysl Šámal. V tomto roce byl zatčen Karel Kramář, který byl v roce 1916 odsouzen za velezradu a vyzvědačství k trestu smrti. Tento trest mu byl později změněn na doživotí a v roce 1917 byl amnestován a propuštěn na svobodu. V procesu byl souzen i další člen maffie novinář Alois Rašín.

Činnost maffie byla ke konci spojena s Národním výborem československým, který vznikl v červnu 1918.

Z uvedeného je zřejmé, že RNDr. Jozef Krajčovič objevil v knize [6] zajímavou chybu, která vznikla z neznalosti našich reálií a kontextu dané doby. Na druhou stranu jsme se díky uvedené knize dostali k informacím o pro mne do té doby „záhadném“ šifrátoru, který pro použití Maffie vyrobil již několikrát zmíněný ing. Baráček.

Pro zajímavost lze uvést ještě další podklady, které se k tomuto šifrátoru (pravděpodobně jednomu z prvních šifrátorů používaných v ČR/SR) vztahují: šifrový text, který se dochoval, a popis rozložení abecedy na kolech šifrátoru.

Čtenář si jistě všimne hned několika zajímavostí. Atypicky je používána abeceda s diakritikou a rozložení znaků ve schématu na obrázku neodpovídá rozložení znaků kol na fotografii. To samozřejmě může vyvolat oprávněné dohady, zda kol nebylo více a zda nebyla dokonce vyměnitelná.



Text zašifrovaný šifrátorem ing.Baráčka / Maffie [8]

Schéma pravděpodobně popisující způsob šifrování a rozložení znaků na kolech šifrátoru ing.Baráčka /Maffie [8]

Na závěr si ještě povšimněme, že počet znaků na obou kolech je shodný. V případě zařízení uloženého v expozici VHÚ v Praze je rozložení znaků na kolech toto:

ABCDEFGHIJKLMNOPRSTUVXYZ123456789-

A9B8C7D6E5F4G3H2Q1IZJYKXLVMUNTOSRP

Literatura

[1] RNDr. Jozef Krajčovič – osobní sdělení

[2] Šklíba, K.: Československé šifrovací stroje z období 1930–1939 a 1945–1955, Crypto-World 78/2007, pp. 2-5.

[3] Vondruška, P.: Vývoj kryptografických zařízení v ČS(S)R , sborník MKB 2009, Praha

[4] Čapek, K.: Hovory s T. G. Masarykem, Československý spisovatel, Praha 1969

[5] Vondruška, P.: Kryptologie, šifrování a tajná písma, Albatros, 2006

[6] Butyrskij, L. S., Larin, D. A., Sankin, G. P.: Kriptograficeskij front Velikoj Otecestvennoj , Moskva, Gelios ARV, Moskva 2012

[7] Maffie [online] URL:< http://cs.wikipedia.org/wiki/Maffie >

[8] Seriál České televize a Karla Pacnera: Československo ve zvláštních službách, epizoda č.2, Jak jsme bourali monarchii, 2002

D. Call for Papers
Mikulášská kryptobesídka

27. – 28. listopad 2014, Praha
http://mkb.tns.cz

Základní informace

Mikulášské kryptobesídky už letos bude dva kusy po tuctu. Je zaměřena na podporu úzké spolupráce odborníků se zájmem o teoretickou a aplikovanou kryptografii a další příbuzné oblasti informační bezpečnosti. Hlavním cílem je vytvořit prostředí pro neformální výměnu informací a nápadů z minulých, současných i budoucích projektů. Cítíme potřebu setkání expertů s jejich kolegy bez obchodních vlivů, starostí s (potenciálními) zákazníky, šéfy a dalšími rozptylujícími faktory. ;-)

Workshop se skládá ze (a) dne prezentací příspěvků, diskusí a neformálního setkání ve čtvrtek *27. listopadu* a (b) půldne prezentací příspěvků a diskusí v pátek *28. listopadu 2014*. Pro workshop jsou domluveny zvané příspěvky od:

- Joachim Posegga: Alice in the Cloud: Insights on Security of Air Traffic Control Communication.
- Gregor Leander: Lightweight Cryptography.
- Karthik Bhargavan: Breaking and Fixing the TLS Cryptographic Protocol.
- Karsten Nohl: Bude potvrzeno koncem srpna.
- Peter Gazi: Key-Length Extension for Block Ciphers: Plain and Randomized Cascades.

Podrobné informace, včetně pokynů k registraci, se budou průběžně objevovat na www stránkách workshopu: http://mkb.tns.cz.

Pokyny pro autory

Přijímány jsou příspěvky zaměřené především na oblasti kryptoanalýzy, aplikované kryptografie, bezpečnostních aplikací kryptografie a dalších souvisejících oblastí. **Návrhy se přijímají odděleně pro sekci KEYMAKER (studentská soutěž) a pro hlavní program workshopu.** Oba druhy návrhů mají požadovaný rozsah 5-15 stran A4 a připravenost pro anonymní hodnocení (bez jmen autorů a zjevných odkazů). Identifikační a kontaktní údaje prosím pošlete v těle e-mailu s příspěvkem jakožto přílohou a jasným označením KEYMAKER, nebo STANDARD TRACK.

Šablony pro formátování příspěvků pro Word a LaTeX lze získat na www stránkách workshopu: http://mkb.tns.cz. Příspěvky mohou být napsané v češtině, slovenštině, nebo angličtině.

Příspěvky připravené podle výše uvedených pokynů zasílejte ve formátu RTF, nebo PDF a tak, aby přišly nejpozději do *30. září 2014*. Pro podávání příspěvků prosím použijte adresu matyas ZAVINAC fi.muni.cz a do předmětu zprávy uveďte „MKB 2014 – navrh prispevku“. Příjem návrhů bude potvrzován do dvou pracovních dnů od přijetí.

Návrhy příspěvků budou posouzeny PV a autoři budou informováni o přijetí/odmítnutí do *30. října*. Příspěvek pro sborník workshopu pak musí být dodán do *11. listopadu*.

Důležité termíny

Návrhy příspěvků:	30. září 2014
Oznámení o přijetí/odmítnutí:	30. října 2014
Příspěvky pro sborník:	11. listopadu 2014
Konání MKB 2014:	27. – 28. listopadu 2014

Programový výbor

Michal Hojsík, Honeywell a MFF UK, Praha, CZ
Marek Kumpošt, NetSuite & FI MU, Brno, CZ
Vašek Matyáš, FI MU, Brno, CZ – předseda
Tomáš Rosa, Raiffeisenbank a UK, CZ
Luděk Smolík, Siegen, DE
Martin Stanek, UK, Bratislava, SK
Pavol Zajac, STU, Bratislava, SK

Mediální partneři

E. O čem jsme psali v předchozích 151 číslech…

Kompletní obsah všech **151** dosud vyšlých čísel od roku 1999 je dostupný zde:
http://crypto-world.info/index2.php?vyber=obsah
http://crypto-world.info/obsah/obsah_roky.pdf

Přehled obsahu posledních vydaných čísel

Crypto-World 5-6/2013

Crypto-World 7-8/2013

Crypto-World 9-10/2013

Příloha: ukázka z knihy Aplikovaná kryptografie
http://crypto-world.info/casop15/Burdaákryptografie.pdf

Crypto-World 11-12/2013

Příloha k článku D: http://web.telecom.cz/depotpv/ASD12/priloha_k_D.zip

Crypto-World 1-3/2014

Crypto-World 4-5/2014

Příloha: CFP_MKB2014.pdf http://crypto-world.info/casop16/CFP_MKB2014.pdf

F. Závěrečné informace

1. Sešit

Crypto-World je oficiální informační sešit "***Kryptologické sekce Jednoty českých matematiků a fyziků***" (GCUCMP). Obsahuje články podepsané autory. Případné chyby a nepřesnosti jsou dílem autorů jednotlivých podepsaných článků, GCUCMP za ně nemá odbornou ani jinou zodpovědnost.
Adresa URL, na níž můžete najít tento sešit (zpravidla 3 týdny po jeho rozeslání) a předchozí e-ziny, denně aktualizované novinky z kryptologie a informační bezpečnosti, normy, standardy, stránky některých členů a další související materiály: http://crypto-world.info

2. Registrace / zrušení registrace

Zájemci o e-zin se mohou zaregistrovat pomocí e-mailu na adrese pavel.vondruska@crypto-world.info (předmět: Crypto-World) nebo použít k odeslání žádosti o registraci elektronický formulář na http://crypto-world.info. Při registraci vyžadujeme pouze jméno a příjmení, titul, pracoviště (není podmínkou) a e-mail adresu určenou k zasílání kódů ke stažení sešitu.
Ke zrušení registrace stačí zaslat krátkou zprávu na e-mail pavel.vondruska@crypto-world.info (předmět: ruším odběr Crypto-Worldu!) nebo opět použít formulář na http://crypto-world.info. Ve zprávě prosím uveďte jméno a příjmení a e-mail adresu, na kterou byly kódy zasílány.

3. Redakce

E-zin Crypto-World

Redakční práce:	Pavel Vondruška
	Jozef Krajčovič
	Jozef Martin Kollar
	Vlastimil Klíma
Přehled autorů:	http://crypto-world.info/obsah/autori.pdf
Webmaster	Pavel Vondruška, jr.

4. Spojení (abecedně)

redakce e-zinu	ezin@crypto-world.info ,	http://crypto-world.info
Vlastimil Klíma	v.klima@volny.cz ,	http://cryptography.hyperlink.cz/
Jozef Martin Kollar	jmkollar@math.sk ,	
Jozef Krajčovič	kryptosvet@gmail.com ,	http://katkryptolog.blogspot.sk
Jaroslav Pinkava	jaroslav.pinkava@gmail.com ,	http://crypto-world.info/pinkava/
Pavel Vondruška	pavel.vondruska@crypto-world.info	http://crypto-world.info/vondruska/index.php
Pavel Vondruška, jr.	pavel@crypto-world.info ,	http://www.pavelvondruska.cz/