Lorenzova šifra

Polona Bogataj

14. junij 2010

1 Uvod

Med 2. svetovno vojno je nemška vojska razvila več kot tisoč različnih šifrirnih naprav, ki so za šifriranje uporabljale ožičene rotorje in so tako lahko generirale na miljone permutacij. To je bilo preveč obsežno za ročno razbijanje šifer in zato so zavezniki zgradili naprave, ki so razbijale te šifre.

Po invaziji na Rusijo junija 1941 se je na nemškem vojaškem operativnem radiju začela pojavljati nova vrsta signalov, poslanih v 5 bitni binarni kodi. Britanci so te signale poimenovali "Tunny", kasneje pa so ugotovili, da so bili generirani s šifrirnim strojem tovarne Lorenz z oznako SZ 40. Do konca vojne so razbijalci kode v Bletchley Parku na mesec dešifrirali na stotine sporočil kriptiranih z Lorenzovo šifro, saj je Tunny direktno povezoval Berlin s sedeži vojaških poveljnikov in so sporočila, ki so jih dešifrirali v Bletchley Parku razkrila strategije in namere na najvišjih nivojih nemške vojske. V ta namen je v Bletchley Parku delovalo na stotine moških in žensk, pomagalo pa jim je tudi 10 računalnikov, ki so jih razvili prav v ta namen in so se imenovali Colossus.

Rzbitje Lorenzove šifre je pomenilo preboj računalništva in matematične analize v kriptografijo in s tem začetek novega obdobja v kriptografiji. Da je to eden izmed zelo pomembnih dosežkov znanosti nam pove tudi dejstvo, da so kljub temu, da je bila uničena celotna strojna oprema in vsi načrti za Colossuse, leta 2007 dokončali delujoč dvojnik Colossusa Mark II. Ogledamo si ga lahko v muzeju Bletchley Park [4].

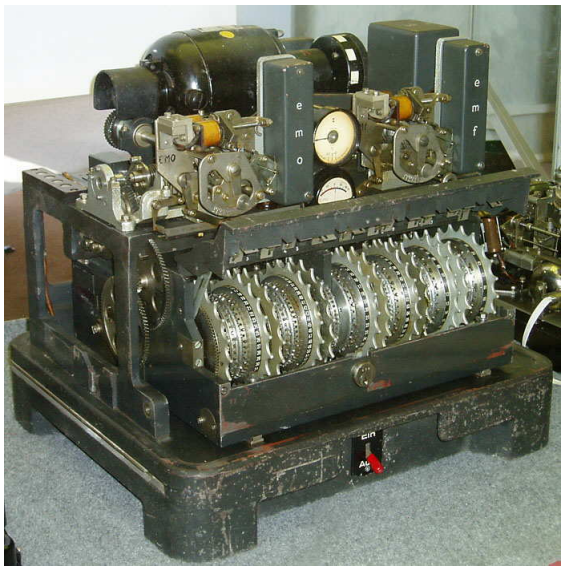
V svoji projektni nalogi bom predstavila Lorenzovo šifro: v prvem delu bom predstavila zgradbo Lorenzovega stroja in korake kodiranja, v drugem delu teoretično podlago za razbijanje tajnopisov in Delta metodo. V zadnjem delu pa bom predstavila algoritem, s katerim lahko razbijemo Lorenzovo šifro. Algoritem je prirejen za današnje računalnike in opravlja isto nalogo kot jo je za potrebe Bletchley Parka opravljal Colossus.

2 Lorenzov stroj

Za imenom Lorenzov stroj se skrivata dve nemški šifrirni napravi: Lorenz SZ40 in Lorenz SZ42, ki jih je med 2. svetovno vojno izdelovalo nemško podjetje Lorenz. Kratica SZ izhaja iz nemške besede "**S**chlüssel**z**usatz", ki pomeni "dodatek za šifriranje" in nakazuje na način šifriranja.

Lorenzov stroj so v nasprotju z Enigmo, ki se je uporabljala na bojiščih, uporabljali za komunikacije na visokih nivojih - tu so lahko uporabljali težke

teleprinterske stroje. Sam Lorenzov stroj je bil velik $51 \times 46 \times 46$ cm in je služil kot dodatek k standardnemu Lorenzovemu teleprinterju.



Slika 1: Ohranjen Lorenzov stroj

Kot zanimivost povejmo, da britanski razbijalci šifer do konca vojne niso videli Lorenzovega stroja, sporočila, ki so bila šifrirana z njimi, pa so razbijali kar dve leti in pol.

V tem razdelku bom najprej predstavila Baudotov kod, na katerem temelji delovanje Lorenzovega stroja, nato bom predstavila zgradbo stroja in korake šifriranja.

2.1 Baudotov kod

Lorenzov stroj je temeljil na Vernamovem kriptosistemu in standardni mednarodni kodi teleprinterjev, kjer je bil vsak simbol besedila pretvorjen v skupino petih električnih impulzov, sestavljenih iz znamenj ($\times$) in presledkov ($\bullet$). Njihovo drugo ime je **Baudotov kod**. Danes lahko to enostavno predstavimo v binarnem zapisu: npr. črka `H`, ki je predstavljena s $\times \bullet \times \bullet\bullet$, bi bila v binarnem zapisu 10100.

V tabeli 1 je predstavljena teleprinterska abeceda. Posebej zanimivi so nekateri posebni znaki:

- `CR` - `carriage return` (pomik na začetek vrste),
- `LF` - `line feed` (pomik v novo vrstico),
- `LTRS` - `letter shift`,
- `FIGS` - `figure shift`,
- `SP` - presledek,
- `BELL` - zvočno opozorilo,

binarni zapis	LTRS	FIGS	binarni zapis	LTRS	FIGS
00011	A	-	10111	Q	1
11001	B	?	01010	R	4
01110	C	:	00101	S	’
01001	D	ENQ	10000	T	5
00001	E	3	00111	U	7
01101	F		11110	V	;
11010	G		10011	W	2
10100	H		11101	X	/
00110	I	8	10101	Y	6
01011	J	BELL	10001	Z	+
01111	K	(	01000	CR	CR
10010	L	)	00010	LF	LF
11100	M	.	00100	SP	SP
01100	N	,	11111	LTRS	LTRS
11000	O	9	11011	FIGS	FIGS
10110	P	0	00000	null	null

Tabela 1: Tabela predstavlja teleprintersko abecedo (prazni simboli pomenijo nedefiniran simbol)

- `ENQ` - `Who are you?` (druga naprava odgovori na ta ukaz),

- `null`.

Vernamov kriptosistem je preprost, saj temelji na operaciji XOR. Tako je ponovno prištevanje ključa izničilo šifriranje in prejemnik je z lahkoto prebral prejeto sporočilo.

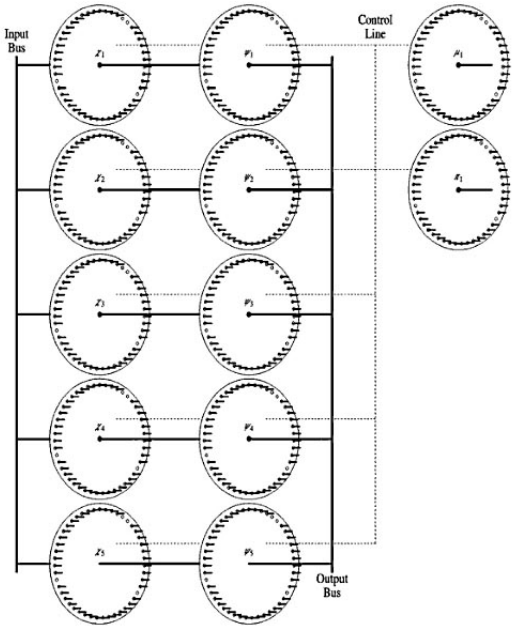
Vernam je predlagal, da bi bili ključi generirani povsem naključno in bi bili predpripravljeni na papirnem traku. Z njega bi se sinhrono s čistopisom bralo simbol za simbolom. Tak kriptosistem (poznan kot **Vernamov enkratni ščit**), ki uporablja popolnoma naključna števila, ima lastnost popolne tajnosti in je nezlomljiv.

V vojnih časih pa nastopi velik problem, kako zagotoviti, da sta na obeh koncih komunikacijskega kanala enaka trakova z naključnimi simboli in da sta oba nastavljena na isti začetni položaj. Družba Lorenz, ki je izdelovala naprave, se je odločila, da je lažje, če zgradijo stroj, ki bo generiral skrivno zaporedje simbolov. Ker pa je naključnost generiral stroj, gre za **psevdonaključno zaporedje simbolov**, in s to odločitvijo je bila storjena napaka, ki je omogočila dešifriranje sporočil šifriranih z Lorenzovo napravo.

2.2 Zgradba Lorenzovega stroja

Pri Lorenzovemu stroju so trakove nadomestila kolesa, ki so generirala periodična zaporedja ničel in enic. Vsako kolo je vsebovalo zaponke, ki so bile enakomerno razporejene na obodu kolesa.

Lorenzov stroj je imel 12 koles (prikazana so na sliki 2, ki jih lahko razdelimo v dve skupini: **šifrirna kolesa**, ki so šifrirala čistopis, in **motorna kolesa**, ki so uravnavala premikanje šifrirnih koles. K šifrirnim kolesom spada 5 χ-koles in 5 ψ-koles, μ in π-kolo pa spadata med motorna kolesa.



Slika 2: Razporeditev koles

Šifrirna kolesa so delovala na naslednji način: ko je bila zaponka aktivna, je kolo svojemu vhodu prištelo 1, ko pa je bila zaponka neaktivna, je kolo svojemu vhodu prištelo 0. Šifrirna kolesa so imela različna števila zaponk (T_i):

i	χ_1	χ_2	χ_3	χ_4	χ_5	ψ_1	ψ_2	ψ_3	ψ_4	ψ_5
T_i	41	31	29	26	23	43	47	51	53	59

Motorna kolesa sta imeli različno število zaponk: μ-kolo je imelo 37 zaponk, π-kolo 61 zaponk, ki so bile lahko aktivne ali neaktivne. Aktivne zaponke bomo označili z $\times$ oz. 1, neaktivne pa z $\bullet$ oz. 0. π-kolo pa se je premaknilo naprej za vsak šifriran simbol, μ-kolo pa se je premaknilo za en položaj naprej le tedaj, ko je bila zaponka na π-kolesu aktivna, sicer pa je ostalo na istem položaju. Tako razširjeno premikanje μ-kolesa imenujemo **osnovni motor**.

Motorna kolesa so uravnavala premikanje šifrirnih koles na naslednji način: ko je bila zaponka na trenutnem položaju μ-kolesa aktivna, so se vsa ψ-kolesa usklajeno premaknila za en položaj naprej, sicer so ostala na istem položaju. χ-kolesa niso bila odvisna od motornih koles in so se za vsak šifriran simbol usklajeno premaknila za en položaj naprej.

Kasneje so osnovnemu šifriranju Lorenzovega stroja dodali tudi **omejitev**, ki je predstavljala še dodatno oviro pri razbijanju prestreženih tajnopisov. Najprej si poglejmo kratek primer izračuna vzorca osnovnega motorja:

Primer. Da bomo lažje določili simbole osnovnega motorja, bomo vzorec π-kolesa oštevilčimo tako, da se številka ponovi vsakič, ko imamo v vzorcu kolesa $\bullet$. Vzorec μ-koles pa oštevilčimo brez posebnosti.

π	$\times$	$\bullet$	$\times$	$\times$	$\times$	$\bullet$	$\times$	$\times$
	1	2	2	3	4	5	5	6
μ	$\times$	$\bullet$	$\bullet$	$\times$	$\bullet$	$\bullet$	$\times$	$\times$
	1	2	3	4	5	6	7	8
Osnovni motor	$\times$	$\bullet$	$\bullet$	$\bullet$	$\times$	$\bullet$	$\bullet$	$\bullet$
	1	2	2	3	4	5	5	6

Aktivni simbol osnovnega motorja skupaj z aktivnim simbolom omejitve določa simbol **skupnega motorja**, ki v resnici določa premikanje ψ-koles. Tudi omejitev je sestavljena iz zaporedja $\times$ in $\bullet$. Vzorec skupnega motorja je določen z naslednjim predpisom: aktivni simbol skupnega motorja je enak $\bullet$ natanko tedaj, ko je aktivni simbol osnovnega motorja enak $\bullet$ in aktivni simbol omejitve enak $\times$. V vseh ostalih primerih je aktivni simbol skupnega motorja enak $\times$.

Primer.

Osnovni motor	$\times$	$\bullet$	$\bullet$	$\bullet$	$\times$	$\bullet$	$\bullet$	$\bullet$
Omejitev	$\bullet$	$\times$	$\bullet$	$\times$	$\times$	$\bullet$	$\bullet$	$\times$
Skupni motor	$\times$	$\bullet$	$\times$	$\bullet$	$\times$	$\times$	$\times$	$\bullet$

Omejitev dobimo iz vzorcev ostalih koles ali čistopisa in ni generirana neodvisno. Poznamo več metod, s katerimi izračunamo vzorec omejitve, vendar se bomo mi omejili le na **omejitev** $\overline{\chi}_2$: aktivni simbol omejitve na kateremkoli položaju je definiran s prejšnjim aktivnim simbolom na kolesu χ_2.

Primer.

χ_2	$\times$	$\times$	$\bullet$	$\bullet$	$\times$	$\times$	$\bullet$	$\times$
Omejitev $\overline{\chi_2}$	$\times$	$\times$	$\times$	$\bullet$	$\bullet$	$\times$	$\times$	$\bullet$

2.2.1 Ključ

Katere zaponke na kolesih so bile aktivne in katere neaktivne je določal ključ, ki je imel dve komponenti:

- 501 bit, ki je določal zaponke vseh 12 koles
- začetne položaje petih χ-koles, petih ψ-koles, μ-kolesa in π-kolesa

Prva komponenta ključa se je spremenila enkrat mesečno, druga komponenta pa bi se morala spremeniti ob vsakem sporočilu. Prvotno so se vsa sporočila začela z nešifriranim indikatorjem, sestavljenim iz dvanajstih črk (npr. `HQIBPEXEZMUG`). Vsaka črka je pomenila začetno nastavitev enega izmed dvanajstih koles in se je pretvorila v število med 0 in 25. To je seveda pomenilo, da niso bile možne vse začetne nastavitve. Pozneje so indikator zamenjali z besedo v knjigi šifer, ki so jo dešifrirali v začetne položaje koles.

2.3 Kodiranje s SZ40

SZ40 in SZ42 sta uporabljala posplošen Vernam-Vigenerjev sistem šifriranja. SZ40 je šifriral po naslednji enačbi:

$$y = x + k \pmod 2,$$

kjer je

- čistopis $\{\underline{x}(j) : \underline{x}(j) = (x_1(j), x_2(j), x_3(j), x_4(j), x_5(j)), j = 0, 1, \ldots\}$ alfanumerično besedilo zašifrirano v 5-bitni niz,

- ključ $\{\underline{k}(j) : \underline{k}(j) = (k_1(j), k_2(j), k_3(j), k_4(j), k_5(j)), j = 0, 1, \ldots\}$ zaporedje 5-bitnih nizov in

- tajnopis $\{\underline{y}(j) : \underline{y}(j) = (y_1(j), y_2(j), y_3(j), y_4(j), y_5(j)), j = 0, 1, \ldots\}$ vsota besedila in ključa po modulu 2.

2.3.1 Koraki kodiranja

Da bomo lahko natančno določili šifrirni proces, moramo vpeljati še nekaj oznak. Naj imajo položaji koles za šifriranje j-te črke čistopisa naslednje oznake:

- $p_i[j]$ naj bo položaj i-tega χ-kolesa

- $q_i[j]$ naj bo položaj i-tega ψ-kolesa

- $u[j]$ naj bo položaj μ-kolesa

- $v[j]$ naj bo položaj π-kolesa

Poglejmo kako poteka kodiranje:

1. j-ti simbol čistopisa je bil zakodiran s pomočjo Baudotovega koda:

$$x(j) \to \underline{x}(j) \equiv (x_1(j), x_2(j), x_3(j), x_4(j), x_5(j)).$$

2. Naj bo $\underline{\chi}(j) \equiv (\chi_1(j), \chi_2(j), \chi_3(j), \chi_4(j), \chi_5(j))$ trenutni izhod χ-koles, ki ga po modulu 2 prištejemo $\underline{x}(j)$ in dobimo vmesni tajnopis $\underline{\tilde{x}}(j) \equiv (\tilde{x}_1(j), \tilde{x}_2(j), \tilde{x}_3(j), \tilde{x}_4(j), \tilde{x}_5(j))$:

$$\underline{x}(j) \to \underline{\tilde{x}}(j) = \underline{x}(j) + \underline{\chi}(j) \pmod 2$$

3. Naj bo $\underline{\psi} = (\psi_1(j), \psi_2(j), \psi_3(j), \psi_4(j), \psi_5(j))$ trenutni izhod ψ-koles, ki ga po modulu 2 prištejemo vmesnemu tajnopisu $\underline{\tilde{x}}(j)$ in dobimo tajnopis $\underline{y}(j) \equiv (y_1(j), y_2(j), y_3(j), y_4(j), y_5(j))$:

$$\underline{\tilde{x}}(j) \to \underline{y}(j) = \underline{\tilde{x}}(j) + \underline{\psi}(j) \pmod 2$$

4. Nekatera kolesa se premaknejo:

   (a) Vsa χ-kolesa se premaknejo za en položaj v nasprotni smeri urinega kazalca:
   $$p_i[j+1] = p_i[j] + 1 \pmod{T_i}$$

   (b) Vsa ψ-kolesa se premaknejo za en položaj v nasprotni smeri urinega kazalca, če je trenutni izhod $\mu(q[j])$ μ-koles enak 1:
   $$q_i[j+1] = (q_i[j] + \mu(u[j])) \pmod{S_i}$$

   (c) μ-kolo se zavrti za en položaj v nasprotni smeri urinega kazalca, če je trenutni izhod π-kolesa 1:
   $$u[j+1] = (u[j] + \pi(v[j])) \pmod{37}$$

(d) π-kolo se zavrti za en položaj v nasprotni smeri urinega kazalca:

$$v[j+1] = (v[j]+1) \pmod{61}$$

Pravila lahko zapišemo tudi drugače:

položaj π-kolesa	$v[j] = j \pmod{61}$
položaj μ-kolesa	$u[j] = [u[j-1] + \pi(v[j-1])] \pmod{37}$
položaj χ-koles	$p_i[j] = j \pmod{T_i}$
položaj ψ-koles	$q_i[j] = [q_i[j-1] + \mu(u[j-1])] \pmod{S_i}$

Če zapišemo enačbe, ki opisujejo postopek šifriranja j-tega simbola (velja $1 \leq i \leq 5$):

$$y_i(j) = x_i(j) + k_i(j) \tag{1}$$
$$k_i(j) = \chi_i(p_i[j]) + \psi_i(q_i[j]) \tag{2}$$

Primer. Beseda `KONEC` se zašifrira takole:

	χ-kolesa			ψ-kolesa		
x_i	Baudot	Zaponke	Izhod	Zaponke	Izhod	y_i
K	01111	10101	11010	01101	10111	Q
O	11000	11111	00111	00100	00011	A
N	01100	10111	11011	11000	00011	A
E	00001	01100	01101	10100	11001	B
C	01110	01011	00101	01111	01010	R

3 Razbijanje tajnopisov

Pri razbijanju tajnopisov ločimo več problemov.

Problem 1. ***Vhod:*** *tajnopis y*
Izhod: *vzorci posameznih koles in njihovi začetni položaji, čistopis x*

Problem 2. ***Vhod:*** *tajnopis y, vzorci vseh koles*
Izhod: *začetni položaji koles, čistopis x*

V svoji projektni nalogi se bom omejila na reševanje drugega problema.

Glavna težava pri iskanju pravilnih začetnih nastavitev koles Lorenzovega stroja je gotovo število vseh možnih kombinacij, ki bi jih bilo potrebno preveriti pri napadu z grobo silo:

$$41 \cdot 31 \cdot 29 \cdot 26 \cdot 23 \cdot 43 \cdot 47 \cdot 51 \cdot 53 \cdot 59 \cdot 61 \cdot 37 \approx 1.6 \cdot 10^{19}$$

To pomeni, da bi ob predpostavki, da bi lahko naredili 1000 testov na sekundo, za pregled vseh kombinacij porabili 500 milijonov let. Zato je očitno potrebno poiskati bolj prefinjen pristop k reševanju zastavljenega problema.

Razbijalci šifer v Bletchley Parku so uspešno razbili nekaj začetnih tajnopisov in ta sporočila so služila za raziskovanje lastnosti Lorenzovih strojev. Tako so odkrili, da je besedilo vsebovalo več parov ponovljenih simbolov, saj so nemški

operaterji pogosto ponavljali kontrolne simbole (v tabeli 1 so v stolpcu LTRS) in s tem preprečili njihovo izgubo med oddajanjem, saj to bi pomenilo niz napak v nadaljnjem delu besedila. Tako bi na primer besedilo `LUFTWAFFE FLTGR (ROEM XVI)` oddajali kot `88LUFTWAFFE9FLTGR9++K88ROEM9XVI++L88`.

Tako je bila glavna naloga razbijalcev šifer sestavljanje procedure, ki bo omogočala avtomatsko prepoznavanje podvojenih simbolov z napravo. Prvo so za to nalogo sestavili Heath Robinson, ki ga je kasneje nasledil Colossus. Delovanje obeh je temeljilo na Delta metodi, ki jo bom podrobneje predstavila v nadaljevanju tega razdelka. Na koncu razdelka bom predstavila kako je iskanje potekalo v praksi in koliko simbolov tajnopisa potrebujemo za uspešnost Delta metode.

3.1 Delta metoda

Naj bo x tajnopis in x' sporočilo, v katerem so simboli premaknjeni za eno mesto v desno:

x `9FLTGR9++K88ROEM9XVI++L88`
x' `9FLTGR9++K88ROEM9XVI++L88`

Navpične pare simbolov v zgornjih dveh besedilih lahko seštejemo po modulu 2. Tako na primer dobimo naslednja seštevanja:

`9`	=	0	0	1	0	0
`R`	=	0	1	0	1	0
		0	1	1	1	0

`+`	=	1	1	0	1	1
`+`	=	1	1	0	1	1
		0	0	0	0	0

Naj bo Δx rezultat tega seštevanja po modulu. Če to zapišemo bolj natančno: n-ti simbol v Δx je vsota simbolov na mestih n in $(n-1)$ v x.

Za zgornji primer to pomeni[1]:

x `9FLTGR9++K88ROEM9XVI++L88`
Δx `D84RTC8/HT/YLBXOBAOX/DF/`

Opazimo lahko, da se za vsak ponovljen simbol v x, v Δx pojavi simbol `/`. To je pomembna lastnost Δx, ki je odločilno vplivala na razbijanje Lorenzove šifre. Kasneje so se tako v večini primerov uporabljali ti. delta simboli (npr. Δy, Δx, ...) namesto originalnih simbolov (npr. y, x, ...).

Če združimo osnovni enačbi 1 in 2 za šifriranje j-tega simbola, dobimo

$$y_i(j) = x_i(j) + \chi_i(p_i[j]) + \psi_i(q_i[j])$$

oz. če zapišemo bolj splošno:

$$y_i = x_i + \chi_i + \psi_i \tag{3}$$

Enačbi 3 lahko na obeh straneh po modulu 2 prištejemo χ_i in dobimo

$$y_i + \chi_i = x_i + \psi_i. \tag{4}$$

Naj bo $d_i = y_i + \chi_i$ in tako lahko enačbo 4 zapišemo kot

$$d_i = x_i + \psi_i.$$

[1] Zaradi preprostejšega zapisa bom 00000 namesto z `null` označevala z /.

Zapišemo lahko tudi ustrezni delta enačbi:

$$\Delta d_i = \Delta y_i + \Delta \chi_i \tag{5}$$
$$\Delta d_i = \Delta x_i + \Delta \psi_i. \tag{6}$$

S kombinacijo zgornjih enačb in analize verjetnosti dobimo učinkovite metode za iskanje začetnih položajev χ-koles. Logično je, da začnemo z iskanjem začetnega položaja enega χ-kolesa, saj bi to pomenilo, da bi morali pregledati samo T_i različnih možnosti (za 1. kolo bi tako morali pregledati le 41 možnosti).

Najprej bomo vpeljali novo oznako: ψ_i' naj označuje razširjen ψ_i (razširjen ψ_i upošteva tudi premikanje kolesa glede na μ-kolo).

Izračunajmo verjetnost, da najdemo začetni položaj kolesa χ_1: naj bo $P[\Delta x_1 = 0] = p$. Zaradi prisotnosti ponovljenih simbolov (in posledično znakov $/= 00000$ v Δx) pričakujemo, da bo ta verjetnost večja od $\frac{1}{2}$.

Naj bo $P[\psi\text{-kolesa se premaknejo}] = a$ in $P[\Delta\psi_1 = 1] = b$. Da se zgodi dogodek $\psi_1' = 1$ se morata zgoditi dva dogodka:

1. ψ-kolesa se morajo premakniti in
2. $\Delta\psi_1 = 1$.

Če se ψ-kolesa ne bi premaknila, potem $\Delta\psi_1'$ ne bi mogel biti enak 1. Tako je verjetnost enaka

$$P[\Delta\psi_1' = 1] = ab \tag{7}$$

Če uporabimo enačbo (6), dobimo da je verjetnost dogodka $\Delta d_1 = 0$ enaka

$$\begin{aligned} P[\Delta d_1 = 0] &= P[\Delta x_1 = 0] \cdot P[\Delta\psi_1' = 0] + P[\Delta x_1 = 1] \cdot P[\Delta\psi_1' = 1] \\ &= p(1-ab) + (1-p)ab = p - 2pab + ab \\ &= p + ab(1-2p) \end{aligned}$$

Naj bo $u = p + ab(1-2p)$. Če bi veljalo, da je $u > \frac{1}{2}$, bi to pomenilo, da ob pravilnem začetnem položaju kolesa, dogodek $\Delta d_1 = 0$ ni naključen. Tako bi lahko našli pravilni začetni položaj kolesa χ_1 s štetjem 0 v dovolj dolgem nizu Δd_1 bitov, ki smo ga dobili iz celotnega tajnopisa. Štetje bi bilo potrebno ponoviti na vseh možnih začetnih položajih kolesa χ_1. Pravilni začetni položaj bi imel največji seštevek 0, kajti seštevki na preostalih začetnih položajih bi bili povezani z naključno verjetnostjo $\frac{1}{2}$. Za štetje simbolov bi uporabili zvezo (5) (Δy_1 in $\Delta\chi_1$ poznamo).

Vendar pa je nemška vojska marca leta 1942 vpeljala pravilo $a \cdot b = \frac{1}{2}$ na vseh kolesnih vzorcih. Tako je vrednost u-ja enaka

$$u = p + ab(1-2p) = \frac{1}{2}$$

in je bilo zaporedje Δd_1 bitov dobljeno iz enega bitnega niza navidezno naključno ne glede na vrednost p ter je bilo tako nemogoče poiskati pravilne začetne nastavitve za eno kolo.

Seveda je naslednja ločina ideja, da poiščemo začetna položaja para koles (npr. χ_1 in χ_2). Pregledati bi morali več možnih parov:

$$41 \cdot 31 = 1271.$$

Poglejmo verjetnostno analizo za par koles (gledamo prvi in drugi bitni niz): naj bo

$$P[\Delta\psi_1 = 1] = b_1 \text{ in } P[\Delta\psi_2 = 1] = b_2.$$

Nemško pravilo $a \cdot b = \frac{1}{2}$ je v tem primeru odpovedalo: ker je vrednost a fiksna, velja $b_1 = b_2 = b$:

$$\begin{aligned} P[\Delta\psi_1 + \Delta\psi_2 = 0] &= P[\Delta\psi_1 = 0] \cdot P[\Delta\psi_2 = 0] + P[\Delta\psi_1 = 1] \cdot P[\Delta\psi_2 = 1] \\ &= (1-b)(1-b) + b \cdot b \\ &= (1-b)^2 + b^2 \end{aligned}$$

Poglejmo dogodek $\Delta\psi_1' + \Delta\psi_2' = 0$, ki se zgodi le v dveh primerih:

- ko se ψ-kolesa ne premaknejo (verjetnost tega dogodka je $1 - a$) ali
- ko se premaknejo in velja $\Delta\psi_1 + \Delta\psi_2 = 0$.

Od tod sledi:

$$\begin{aligned} P[\Delta\psi_1' + \Delta\psi_2' = 0] &= (1-a) + a\left((1-b)^2 + b^2\right) \\ &= 1 - 2ab + 2ab^2 \\ &\text{ker velja } a \cdot b = \frac{1}{2}: \\ &= b \end{aligned}$$

Razbijalci šifer v Bletchley Parku so na podlagi znane zgradbe Lorenzovega stroja in vzorcev zank na kolesih domnevali, da je 0.703 realistična ocena za verjetnost dogodka $P[\psi\text{-kolesa se premaknejo}]$. Ker je $ab = \frac{1}{2}$, je $b = \frac{1}{2a} = 0.71$. Tako je

$$P[\Delta\psi_1' + \Delta\psi_2' = 0] = 0.71$$

Prejšna sporočila so pokazala, da je verjetnost

$$P[\Delta x_1 + \Delta x_2 = 0] = 0.6$$

Če združimo enačbo (6) za $i = 1$ in $i = 2$, dobimo:

$$\Delta d_1 + \Delta d_2 = (\Delta x_1 + \Delta x_2) + (\Delta\psi_1 + \Delta\psi_2) \tag{8}$$

Iz tega sledi:

$$\begin{aligned} P[\Delta d_1 + \Delta d_2 = 0] &= P[(\Delta x_1 + \Delta x_2) + (\Delta\psi_1 + \Delta\psi_2) = 0] \\ &= P[(\Delta x_1 + \Delta x_2) = 0] \cdot P[(\Delta\psi_1 + \Delta\psi_2) = 0] + \\ &+ P[(\Delta x_1 + \Delta x_2) = 1] \cdot P[(\Delta\psi_1 + \Delta\psi_2) = 1] \\ &\text{če uporabimo prej podane numerične približke dobimo:} \\ &= 0.6 \cdot 0.7 + (1 - 0.6)(1 - 0.7) \\ &= 0.54 \end{aligned}$$

Ta rezultat nam pove, da vsota dveh bitnih nizov ni naključna in nam pomaga poiskati pravilna začetna položaja dveh χ-koles. Ker pa je razlika med pravilnima začetnima položajema in ostalimi položaji koles majhna, moramo štetje simbolov opraviti na dovolj dolgem tajnopisu.

Če združimo enačbo (5) za $i = 1$ in $i = 2$, dobimo:

$$\Delta d_1 + \Delta d_2 = (\Delta y_1 + \Delta \chi_1) + (\Delta y_2 + \Delta \chi_2) \tag{9}$$

To enačbo pa lahko uporabimo za preštevanje 0 v praksi.

Predvidevajmo, da število preštetih znakov 0 za vsak par napačnih začetnih položajev ustreza binomski verjetnostni porazdelitvi (kjer velja $p = \frac{1}{2}$). Če tajnopis vsebuje N simbolov, potem je pričakovan seštevek, ko imata kolesa χ_1 in χ_2 pravilne začetne položaje, enak $0.54N$. Za vsak par napačnih začetnih položajev je pričakovan naključni seštevek enak $0.5N$ in standardna deviacija je enaka $\sqrt{\frac{N}{2}}$.

Naj bo deviacija od povprečja enaka $0.04N$. Ker želimo, da ima to pomen, to pomeni, da ne sme biti manjša kot izbrani večkratnik standardne deviacije. V Bletchley Parku so za večkratnik izbrali število 4. To pomeni, da mora veljati naslednja neenakost:

$$0.04N > 4 \cdot \sqrt{\frac{N}{2}}$$

Če kvadriramo obe strani neenačbe dobimo:

$$\begin{aligned} (0.04)^2 \cdot N &> 4 \\ N &> 2500 \end{aligned}$$

To nam da minimalno dolžino tajnopisa, ki ga potrebujemo za razlikovanje med pravilnimi in napačnimi začetnimi položaji.

3.2 Kako je potekalo iskanje v praksi?

Štetje znakov 0 se je izvajalo na računalniku Colossus z algoritmom, ki je temeljil na zvezi (9). Stroj je optično bral šifriran bitni niz iz preluknjanega papirnatega traku in ga kombiniral s kolesnimi vzorci χ-koles, kakor je zahteval algoritem. Na koncu procesa je izpisal rezultat.

Število pregledanih bitov je bilo veliko. Če je bila dolžina kriptograma 3000 simbolov, potem je bil zgornji algoritem uporabljen $41 \cdot 31 \cdot 3000 = 3813000$-krat. Da se je to lahko odvijalo v realnem času, je moral biti stroj zelo hiter. Colossus je tako lahko prebral 5000 šifriranih simbol na sekundo in je zgoraj opisani problem rešil v 13 minutah.

3.3 Nemška napaka

Prvič so sporočila nemške vojske, ki so bila šifrirana s strojem Lorenz, prestregli leta 1940. John Tiltman, ki je deloval v Bletchley Parku, je prepoznal, da so bila sporočila šifrirana s pomočjo Vernamovega sistema. To je pomenilo, da bi lahko v primeru, ko bi nemški operatorji uporabili iste nastavitve Lorenzovega stroj za šifriranje dveh sporočil, s seštevanjem obeh prestreženih tajnopisov dešifrirali sporočilo. Število prestreženih sporočil je naraščalo, vendar do 30. avgusta, ko sta nemška operaterja napravila napako, ni bilo vidnejšega napredka.

Ta dan je moral nemški operater poslati dolgo sporočilo (njegova dolžina je bila približno 4000 simbolov) iz enega dela nemškega visokega poveljstva na drugi del - najbrž je bilo sporočilo poslano iz Aten na Dunaj. Pravilno je nastavil Lorenz stroj, poslal 12 črkovni indikator operaterju na drugem koncu.

Drugi operater je nastavil svoj Lorenz stroj in zaprosil prvega operaterja, naj začne oddajati sporočilo. Ko je bilo poslanih skoraj 4000 na roke zašifriranih simbolov, je drugi operater, ki je prejemal besedilo po radiju poslal v neščini sporočilo "Tega nisem dobil, pošlji ponovno".

Nato sta oba operaterja ponovno nastavila Lorenz stroja na prvotno začetno pozicijo, kar je bilo seveda prepovedano. Operater, ki je pošiljal sporočilo je nato znova na roko zakodiral celotno sporočilo, vendar pa je naredil še večjo napako: pri ponovnem kodiranju je okrajšal nekatere besede. Če bi uporabil enako sporočilo kot pri prvem pošiljanju, bi Britanci dobili le dve identični kopiji tajnopisa.

Sporočilo se je začelo z znano nemško frazo `SPRUCHNUMMER` (številka sporočila). Prvič je operater poslal `SPRUCHNUMMER`, drugič pa je skrajšal besedo v `SPRUCHNR` - `NR` je bila okrajšava za `NUMMER`. To je pomenilo, da sta bili sporočili od črke `N` naprej različni. Vendar pa je stroj generiral enako zaporedje ključev in zato sta bila tudi tajnopisa od te točke naprej različna.

Britanski razbijalci šifer so prepoznali, da gre za isto sporočilo (po enakem 12 mestnem indikatorju) in ta "kombinirana napaka" jim je omogočila, da so v celoti dešifrirali oba teksta in razkrili skrivnost Lorenzovega stroja.

3.4 Švedsko razbijanje

Na tem mestu naj omenim še uspešno švedsko razbijanje Lorenzove šifre. Spomladi leta 1940 je priznan švedski kriptoanalitik Arne Beurling razbil Lorenzovo šifro - prisluškovali so za Švedsko zelo pomembni povezavi med Nemčijo in Norveško. Kmalu zatem je L. M. Ericsson zgradil napravo, ki je avtomatsko razbijala Lorenzovo šifro. Z njeno pomočjo je švedska vojska prebirala telegrame do leta 1943, ko so Nemci izboljšali Lorenzove naprave in Beurlingov sistem ni več deloval.

4 Algoritem za razbijanje Lorenzove šifre

Lorenzovo šifro so v Bletchley Parku prvo razbijali na roke s pomočjo ti. Tutteje ve metode, kasneje pa so dešifriranje pohitrili in so zgradili Heath Robinson in Colossus. Njuno delovanje je temeljilo na Newmanovi Delta metodi 3.1. Oba sta elektronsko preštevala simbole in s tem močno pohitrira delo kriptoanalitikov in drugih, ki so sodelovali pri razbijanju sporočil.

V svojem projektu ne bom predstavila algoritma, ki so ga implementirali na Colossusu, saj je bila njegova naloga le preštevanje simbolov, vse nastavitve pa so na roke določali kriptoanalitiki v Bletchley Parku. Predstavljeni algoritem je primeren za današnje računalnike. Algoritem ima kvadratično časovno zahtevnost, ki je odvisna od podanega tajnopisa.

4.1 Algoritem

V algoritmih bomo uporabljali stroj S, z naslednjimi atributi:

- vzorce koles: zaporedje 0 in 1, ki ustrezajo aktivnim in pasivnim zaponkam koles,
- začetne položaje koles `pos`,

- boolean `omejitev`, ki nam pove, ali se upošteva tudi omejitev,

- vzorec `Chi2back`, ki ustreza omejitvi $\overline{\chi_2}$.

Prvo poglejmo algoritem za izračun premikanja ti. motornih koles (μ in π). Algoritem bo sledil postopku, ki je opisan v poglavju 2.2.

Algoritem 1 - `motorna_kolesa` - Glede na vrednosti zaponk koles in omejitve algoritem izračuna kakšno bo gibanje koles. Oznaka OM označuje osnovni motor, SM pa skupni motor.

Vhod: Omejitev, stroj S
  if vrednost trenutne zaponke na μ-kolesu $\neq 0$ **then**
    OM := **false**
  else
    OM := **true**
  end if
  if Omejitev = **true then**
    if S.Chi2Back = 1 **and** OM = **false then**
      SM := **false**
    else
      SM := **true**
    end if
  else
    SM := OM
  end if

Algoritem nima posebne časovne zahtevnosti: $\mathcal{O}(1)$.

Pri iskanju pravilnih začetnih položajev potrebujemo oceno, kako dober je naš trenutni rezultat. Mi si bomo pomagali s funkcijo, ki so jo razvili v Bletchley Parku:

$$\text{Vsota} = \frac{1}{\text{Total}} \cdot \sum_{S \in \text{Symbols}} \frac{\text{Counts}(S)^2}{\text{Probability}(S)} - \text{Total} \tag{10}$$

Algoritem 2 - `izracun` - S pomočjo enačbe (10) izračuna oceno za izbrani tekst.

Vhod: tekst T, verjetnosti P
Izhod: ocena Sum
  for I in T **do**
    $S := T(I)$
    Counts(S) := Counts(S) + 1
  end for
  Total := dolžina T
  for S in Counts **do**
    D:= Counts(S)
    Sum := Sum+$D \cdot D$/Probability(S)
  end for
  return Sum/Total−Total

Total predstavlja število vseh simbolov, množica Symbols vsebuje vse možne simbole, funkcija Count(S) prešteje kolikokrat se v besedilu pojavi simbol S,

funkcija Probability(S) pa nam pove, kakšna je verjetnost pojavitve simbola S v besedilu in jo izračunamo vnaprej.

Kakšna je računska zahtevnost algoritma 2? Naj bo n dolžina teksta T in m število simbolov v množici Symbol. Časovna zahtevnost algoritma je $\mathcal{O}(n+m)$ kar lahko ob realni predpostavki $n > m$ poenostavimo v $\mathcal{O}(n)$.

4.1.1 Algoritma za iskanje pravilnih začetnih nastavitev koles

Za iskanje pravilnih začetnih položajev koles bomo uporabili dve metodi. Ena bo preprosto pregledovanje vseh možnih kombinacij (Algoritem 6), druga pa pregleda le omejeno naključno množico kombinacij (Algoritem 4).

Oba algoritma za iskanje začetnih položajev uporabljata tudi algoritem `kriptiraj`.

Algoritem 3 - `kriptiraj` - Podanemu tajnopisu X ustrezno prišteje ključ K.

Vhod: stroj S, tajnopis X
Izhod: čistopis Y
  for crka in X **do**
    Y(crka) := X(crka) **xor** `kljuc`(S)
    glede na omejitev premakni kolesa stroja S
  end for
  return Y

Funkcija `kljuc`(S) vrne vrednost ključev - pri tem upošteva, ali prištevamo samo χ kolesa, ali tudi ψ kolesa (v tem primeru vrne XOR vrednosti χ in ψ kolesa).

Verjetnostnemu algoritmu `verjetnostno_iskanje` za iskanje pravilnih začetnih položajev podanih koles W podamo maksimalno željeno število iteracij (max). Možne množice koles W so `Chi`= $\{\chi_1, \chi_2, \chi_3, \chi_4, \chi_5\}$, `Psi`= $\{\psi_1, \psi_2, \psi_3, \psi_4, \psi_5\}$ in `Motor`= $\{\mu, \pi\}$. V vsaki iteraciji nato algoritem izbere naključne začetne položaje vseh koles in izračuna oceno.

Tabela vseh možnih začetnih položajev je lahko realizirana kot razpršena tabela (hash tabela), kjer so ključi tabele naraščajoča števila, vsebina elementa pa je sestavljena iz množice začetnih nastavitev in zastavice `obisk`, ki nam pove, ali smo že preverili ta začetni položaj. Tako definirana struktura nam omogoča, da lahko za verjetnostni algoritem ocenimo časovno zahtevnost, vendar si prvo poglejmo primer vnosa v tabeli.

Primer. Primer vnosa v tabeli `tabela_zacetnih` za W=`Chi`:

$$(\text{ključ}, ((\chi_1, \chi_2, \chi_3, \chi_4, \chi_5), \text{obisk}))$$
$$(15, ((1, 0, 4, 17, 2), \text{false})$$

Do tega vnosa v tabeli bi prišli, ko bi funkcija `random` generirala število 15. Ker je `obisk` enak **false**, bi za nov začetni položaj kolesa χ_1 izbrali položaj 1, χ_2 položaj 0, itd.

Časovna zahtevnost algoritma (4) je odvisna predvsem od števila iteracij **while** zanke. Spodnja meja za število iteracij je očitna: če že v prvo izračunamo najvišjo oceno `best_score`, potem je število korakov enako max +1. V najslabšem primeru pa bi morali pregledali celotno tabelo začetnih vrednosti `tabela_zacetnih`.

Naj bo m število elementov v tabeli `tabela_zacetnih` in n dolžina podanega tajnopisa X. Časovna zahtevnost **while** zanke je v najboljšem primeru $\mathcal{O}(n)$ (klica funkcije `izracun` in `kriptiraj`), v najslabšem primeru pa je odvisen od dolžine teksta oz. od števila elementov v tabeli: $\mathcal{O}(m+n)$. Ker je število m ponavadi večje od dolžine tajnopisa (ti so ponavadi dolžine $< 10^4$ simbolov), lahko poenostavimo na $\mathcal{O}(m)$

Časovna zahtevnost algoritma bi tako bila kvadratična: v najboljšem primeru $\mathcal{O}(n \cdot \max)$, v najslabšem primeru pa je $\mathcal{O}(m^2)$.

Algoritem 4 - `verjetnostno_iskanje` - Verjetnostno iskanje začetnih položajev koles.

Vhod: tajnopis X, stroj S, število ponovitev max, verjetnostna razporeditev P, kolesa W, tabela vseh možnih začetnih položajev podanih koles tabela_zacetnih, tabela nastavitev T
Izhod: tabela nastavitev T

```
no := 0
pos := S.pos
best_score := 0
best_pos := pos
while no < max do
  zac := tabela_zacetnih(random)
  while zac.obisk do
    zac := tabela_zacetnih(random)
  end while
  pos := zac.pos
  text := kriptiraj(S, X)
  score := izracun(text, P)
  v T vstavi (pos, score)
  if score > best_score then
    best_score := score
    best_pos := pos
    no := 0
  else
    pos := best_pos
    no := no + 1
  end if
end while
S.pos := pos
```

Pri algoritmu, ki bo pregledal vse možne začetne položaje, potrebujemo algoritem za premikanje koles. Algoritem (5) premakne eno izmed koles za 1 položaj naprej in vrne **false**. Algoritem vrne **true** šele ko pregleda vse možnosti - tj. ko v istem obhodu **for** zanke premakne vsa kolesa na 0. Njegova časovna zahtevnost je odvisna le od števila koles v množici W in je konstantna.

Algoritem 6 `iskanje_groba_sila` pregleda vse možne začetne položaje za podano množico koles W. Algoritem vrne začetne položaje, ki imajo najboljšo oceno po enačbi (10) in so v urejeni tabeli `tabela_ocen` na prvem mestu.

Naj bo n dolžina podanega tajnopisa X, m pa število vseh pregledanih kombinacij začetnih položajev koles. Število iteracij **while** zanke je odvisno od

Algoritem 5 - `premakni_kolesa` - premakne eno izmed koles na naslednji položaj.

Vhod: kolesa W, položaji pos
Izhod: boolean done

```
for kolo in W do
  if pos(kolo) != kolo.length then
    pos(kolo) := pos(kolo)+1
    return false
  else
    pos(kolo) := 0
  end if
end for
return true
```

spremenljivke `done`, ki postane **true** šele ko so pregledane vse kombinacije. To pomeni, da se bo zanka izvedla m-krat. Funkciji `kriptiraj` in `izracun` imata časovno zahtevnost $\mathcal{O}(n)$, zato je časovna zahtevnost algoritma $\mathcal{O}(m \cdot n)$.

Algoritem 6 - `iskanje_groba_sila` - Iskanje začetnih položajev koles z grobo silo.

Vhod: tajnopis X, stroj S, verjetnostna razporeditev P, kolesa W
Izhod: najboljši položaj pos

```
vse začetne položaje koles v podani množici W postavi na 0
done := false
pos := S.pos
T := tabela ocen
while not done do
  S.pos := pos
  text := kriptiraj (S, X)
  score := izracun(text, P)
  v tabela_ocen vstavi (pos, score)
  done := premakni_kolesa(W, pos, done)
end while
pos := tabela_ocen(1).pos
```

4.1.2 Algoritem za razbijanje Lorenzove šifre

V algoritmu (7) bomo združili vse prej opisane algoritme. Prvo bomo naključno izbrali začetne položaje za ψ-kolesa in kolesi μ in π. Nato bomo z 10 različnimi naključno izbranimi začetnimi položaji izvedli verjetnostno iskanje za χ-kolesa. Najbolj verjetni začetni položaji bodo shranjeni v tabeli T_1 na 1. mestu. Za kolesi μ in π bomo pregledali vse možne kombinacije položajev (to je $37 \cdot 61 = 2257$ položajev). Funkcija nam vrne najbolj verjetna začetna položaja za kolesi, ki ju pridružimo že prej dobljenim položajem χ-koles v `pos`. Nazadnje poiščemo še najbolj verjetne položaje ψ-koles in jih dodamo v `pos`.

Na koncu nastavimo stroj S na najbolj verjetno kombinacijo začetnih položajev in odšifriramo tajnopis X.

Algoritem 7 - `iskanje` - Iskanje začetnih položajev koles.

Vhod: tajnopis X, ključ K, stroj$S$, verjetnostna razporeditev P
Izhod: čistopis Y, začetni položaji koles pos
  zgeneriraj `tabela_zacetnih`, T_1, T_2, T_3
  naključno izberi začetni položaj vseh `Psi` koles
  naključno izberi začetni položaj obeh `Motor` koles
  for $i = 0$ to 10 **do**
    naključno izberi začetni položaj vseh `Chi` koles
    T_1 := `verjetnosto_iskanje`(X, S, 1000, P, `Chi`, `tabela_zacetnih`, T_1)
  end for
  pos := $T_1(1)$
  pos := `iskanje_groba_sila`(X, S, P, `Motor`)
  T_3 := `verjetnostno_iskanje`(X, S, 1000, P, `Psi`, `tabela_zacetnih`, T_3)
  pos := $T_3(1)$
  S.pos := pos
  Y := kriptiraj(S, X)

Na časovno zahtevnost algoritma za razbijanje koles vplivata algoritma `verjetnostno_iskanje` in `iskanje_groba_sila`. Oba imata kvadratično časovno zahtevnost, zato je tudi časovna zahtevnost celotnega algoritma kvadratična.

Literatura

[1] Jack Copeland, *COLOSSUS - The Secret of Bletchley Park's Codebreaking Computers*. Oxford University Press, ZDA, 2006.

[2] Alan G. Konheim, *Computer security and cryptography*. Wiley-Interscience, 2007.

[3] Poročila so dostopna na spletnem naslovu:

http://www.codesandciphers.org.uk/

[4] Spletna stran muzeja Bletchley Park: *http://www.bletchleypark.org/*

Na tej spletni strani lahko najdemo tudi članek o Colossus-u in razbinjanju Lorenzove šifre

(*http://www.bletchleypark.org/content/lorenzcipher.pdf*)