

1

2

STEGANOGRRAFIE A DIGITÁLNÍ MÉDIA

3

ANDREW KOZLÍK

4

Toto jsou provizorní skripta k přednášce Steganografie a digitální média na MFf UK v letním semestru akademického roku 2014/15. Témata zde pokrytá tvoří jen polovinu přednášky. Zbytek látky prozatím najdete v kapitolách 1–5, 7 a 11 knihy *Steganography in digital media* [5] od J. Fridrich.

5

6

7

8

1. STEGANOGRRAFIE V PALETOVÝCH OBRÁZCÍCH

9

1.1. ÚVOD

Na úvod připomeňme pár základních poznatků o reprezentaci barev. Barvy jsou reprezentovány jako uspořádané trojice $(r, g, b) \in \{0, 1, \dots, 255\}^3$, kde jednotlivé složky udávají intenzitu červené, zelené a modré. V množině všech barev můžeme měřit podobnost neboli vzdálenost dvou barev pomocí Eukleidovské normy $\|(r_1, g_1, b_1) - (r_2, g_2, b_2)\| = \sqrt{(r_1 - r_2)^2 + (g_1 - g_2)^2 + (b_1 - b_2)^2}$. Množinu všech barev lze uspořádat lexikograficky jako množinu trojic, tj. nejdříve podle intenzity červené složky a v případě, kdy dvě barvy mají stejnou intenzitu červené, uspořádáme je podle intenzity zelené složky a konečně v případě, kdy dvě barvy mají stejnou intenzitu červené i stejnou intenzitu zelené, uspořádáme je podle intenzity modré složky.

Existují i jiné reprezentace barev, které mohou být vhodnější pro měření podobnosti, např. CIELAB. V prostoru CIELAB jsou barvy rozloženy na jasovou složku a dvě barvosné složky. Transformace mezi reprezentacemi RGB a CIELAB je nelineární.

22

1.2. PALETOVÉ OBRÁZKY

Paletový obrázek sestává z palety barev a z matice indexů. Paleta barev je množina $C = \{c_i \mid i \in I\}$ indexovaná čísly $I = \{0, 1, \dots, |C| - 1\}$. Paleta je zpravidla uložena v souboru jako tabulka velikosti $|C| \times 3$ a indexy jsou ukazatele do této tabulky. Matice indexů má rozměry shodné s rozměry obrázku, její složky odpovídají pixelům obrázku. Barva pixelu je tedy určena indexem na příslušné pozici v matici indexů.

Každý paletový obrázek má mnoho různých reprezentací určených uspořádáním palety. Můžeme totiž vzít soubor obsahující paletový obrázek a v tabulce, kde je uložena paleta, zaměnit první dva řádky, tj. první dvě barvy. Aby obrázek jako takový zůstal zachován, musíme odpovídajícím způsobem přečíslovat matici indexů tak, že všechny složky s indexem 0 přepíšeme na 1 a naopak ty s indexem 1 přepíšeme na 0. Kdykoliv budeme hovořit o přeuspořádání palety, máme zároveň na mysli, že se odpovídajícím způsobem přečíslová matice indexů.

Steganografii v paletových obrázcích lze provádět dvěma základními způsoby.

35

36 1. Volbou uspořádání palety. Množinu všech permutací palety S_C lze uspořádat lexi-
37 kograficky a každé permutaci můžeme tedy přiřadit pořadové číslo od 0 do $|C|! - 1$.
38 Každému pořadovému číslu pak přiřadíme nějakou zprávu. V případě palety s 256
39 barvami máme nosič s kapacitou $\log_2 256! \approx 1684$ bitů, což odpovídá například 1,5
40 SMS zprávám.

41 Výhodou tohoto postupu je, že nezanechává žádnou viditelnou stopu v obrázku jako
42 takovém, protože jediné co děláme je, že volíme jednu z mnoha různých ekvivalent-
43 ních reprezentací obrázku. Nevýhodou je jednak omezená kapacita, jednak to, že
44 chaoticky uspořádaná paleta okamžitě budí podezření. Naprostá většina programů
45 totiž při ukládání paletového obrázku setřídí paletu podle jasů, popřípadě podle
46 odstínu, anebo podle četnosti jednotlivých barev v obrázku. Z toho plyne další ne-
47 výhoda, kterou je to, že otevření stegoobrázku a jeho opětovné uložení má zpravidla
48 za následek zničení nesené zprávy.

49 2. Vkládání do matice indexů.

50 (a) Paletu setřídíme podle jasů. Bity zprávy vkládáme do nejnižších bitů jednot-
51 livých indexů v matici. Uspořádání palety podle jasů zajistí, že změna nejníž-
52 šího bitu v indexu příliš neovlivní jas pixelu. Tento postup používá algoritmus
53 EzStego. Jeho nevýhodou je, že změna nejnižšího bitu v indexu může vést k zá-
54 sadní změně odstínu pixelu.

55 (b) Budeme postupovat jako v předchozím případě, ale paletu uspořádáme tak,
56 aby každé dvě po sobě následující barvy v paletě byly co možná nejpodob-
57 nější. Můžeme ji například uspořádat tak, aby součet $\sum_{i=0}^{|C|-2} \|c_i - c_{i+1}\|$ nabýval
58 minimální možné hodnoty. Problém hledání takového uspořádání je shodný
59 s problémem obchodního cestujícího. Jak známo, jedná se o NP-úplný problém.

60 (c) Zvolíme zcela jiný přístup než je vkládání do nejnižšího bitu. Ke každé barvě
61 v paletě přiřadíme hodnotu 0 nebo 1 takovým způsobem, aby nejpodobnější
62 barva v paletě měla opačnou hodnotu. Hodnotu, kterou každé barvě přiřa-
63 zujeme, nazýváme *parita* barvy. Paritu barev tedy volíme tak, aby se mini-
64 malizovala velikost změn vyvolaných vkládáním. O tomto přístupu pojednává
65 následující část.

66 1.3. VKLÁDÁNÍ S OPTIMÁLNÍM PŘÍŘAZENÍM PARITY

67 Zobrazení, které každé barvě přiřazuje paritu, budeme značit p . Dále budeme pracovat se
68 zobrazením f , které každé barvě přiřazuje nejpodobnější barvu v paletě.

69 **Definice.** Nechtě $C = \{c_i \mid i \in I\}$ je paleta a (p, f) je dvojice zobrazení, kde $p : I \rightarrow \{0, 1\}$
70 a $f : I \rightarrow I$. Jestliže pro každé $i \in I$ platí

$$p(i) = 1 - p(f(i)) \quad \text{a} \quad \|c_i - c_{f(i)}\| = \min_{j \in I \setminus \{i\}} \|c_i - c_j\|,$$

71 pak říkáme, že (p, f) je *optimální přiřazení parity* pro paletu C .

72 Optimální přiřazení parity existuje pro každou paletu, není však jednoznačně určeno.
73 Existenci dokážeme v algoritmu 1.3. Nejednoznačnost je zřejmá už z toho, že je-li (p, f)
74 optimální přiřazení parity pro C , pak také (p', f) , kde $p'(i) = 1 - p(i)$ pro všechna $i \in I$,
75 je optimální přiřazení parity pro C . Jinými slovy, všem barvám můžeme přiřadit opačnou
76 paritu než jakou jim přiřazuje p . Mohou však existovat i další optimální přiřazení parity,
77 například taková, která jsou „na půli cesty mezi p a p' “. To znamená, že u některých palet

78 lze p obrátit na určité podmnožině množiny I a získat nové optimální přiřazení parity.
 79 Dalším důvodem nejednoznačnosti může být situace, kdy paleta obsahuje tři nebo více
 80 různých barev, které jsou navzájem stejně vzdálené, čili např. barvy c_1, c_2, c_3 takové, že
 81 $\|c_1 - c_2\| = \|c_2 - c_3\| = \|c_3 - c_1\|$. Stejně tak ani zobrazení f nemusí být jednoznačně určeno,
 82 protože v paletě se může nacházet barva, ke které existují dvě nebo více nejpodobnějších
 83 barev.

84 Nejednoznačnosti optimálního přiřazení parity se můžeme zbavit tím, že na množině
 85 barev zavedeme lineární uspořádání. To pak lze využít k upřednostnění jednoho opti-
 86 málního přiřazení parity před ostatními; viz již zmíněný algoritmus 1.3. Jedním možným
 87 lineárním uspořádáním palety je její uspořádání v souboru samotném, tj. podle indexů.
 88 Jak už jsme zmínili v předchozí části, spoléhat se na toto uspořádání není ideální. Lepším
 89 řešením je uspořádat barvy lexikograficky jakožto třísloužkové vektory. Jediný problém,
 90 který zde může nastat je situace, kdy paleta obsahuje duplicity. Obsahuje-li totiž paleta
 91 dvě stejné barvy s různými indexy, pak nelze jednoznačným způsobem přiřadit jedné z nich
 92 paritu 0 a druhé paritu 1 bez toho, abychom se odkázali na jejich uspořádání v souboru.
 93 Řešením je ztotožnit duplicitní barvy.

94 Než přistoupíme k algoritmu pro sestavení optimálního přiřazení parity, uvedeme pří-
 95 slušné algoritmy vkládání a extrakce.

96 **Algoritmus 1.1** (vkládání s optimálním přiřazením parity).

vstup: nosič $x = (x_1, \dots, x_n) \in I^n$, zpráva $z = (z_1, \dots, z_m) \in \{0, 1\}^m$,
 97 klíč $\pi \in S_n$, optimální přiřazení parity (p, f)
výstup: stegoobjekt $y = (y_1, \dots, y_n) \in I^n$
 1 **for** $i = 1, \dots, n$ **do**
 2 **if** $i > m$ **or** $p(x_{\pi(i)}) = z_i$ **then**
 3 $y_{\pi(i)} := x_{\pi(i)}$
 98 4 **else**
 5 $y_{\pi(i)} := f(x_{\pi(i)})$
 6 **return** y

99 **Algoritmus 1.2** (extrakce s optimálním přiřazením parity).

vstup: stegoobjekt $y = (y_1, \dots, y_n) \in I^n$, klíč $\pi \in S_n$, délka zprávy m ,
 100 optimální přiřazení parity (p, f)
výstup: zpráva $z = (z_1, \dots, z_m) \in \{0, 1\}^m$
 1 **for** $i = 1, \dots, m$ **do**
 101 2 $z_i := p(y_{\pi(i)})$
 3 **return** z

102 Algoritmus vkládání i algoritmus extrakce oba pracují s optimálním přiřazením pa-
 103 rity (p, f) pro daný obrázek. Měli bychom zdůraznit, že pár (p, f) se v rámci komunikace
 104 neposílá, ale odesílatel i příjemce si ho spočítají samostatně ze znalosti palety. Aby ste-
 105 gosystém byl funkční, je nezbytné, aby obě strany dospěly ke stejnému přiřazení parity.
 106 Zároveň bychom uvítali, kdyby se tak stalo i v případě, že při přenosu dojde k pře-
 107 spořádání palety. Následující algoritmus pro sestavení optimálního přiřazení parity tuto
 108 vlastnost splňuje.

109 **Algoritmus 1.3** (optimální přiřazení parity).

vstup: paleta bez duplicit $C = \{c_i \mid i \in I\}$
 110 **výstup:** optimální přiřazení parity (p, f) pro C , nezávislé na uspořádání palety C
 111

1 $E := \{(\|c_i - c_j\|, c_i, c_j) \mid i, j \in I, i \neq j\}$

```

2  while  $E \neq \emptyset$  do
3     $(d, c_i, c_j) := \min_{\text{LEX}} E$ 
4    if  $p(j)$  is undefined then
5       $p(i) := 0$ 
6       $p(j) := 1$ 
7       $f(i) := j$ 
8       $f(j) := i$ 
9       $E := E \setminus (\mathbb{R} \times \{c_i, c_j\} \times C)$ 
10   else
11      $p(i) := 1 - p(j)$ 
12      $f(i) := j$ 
13      $E := E \setminus (\mathbb{R} \times \{c_i\} \times C)$ 
14   return  $(p, f)$ 

```

112 *Důkaz.* Funkce \min_{LEX} vrací lexikograficky minimální prvek množiny, což zde znamená
113 prvek $(\|c_i - c_j\|, c_i, c_j)$ z množiny E s minimální vzdáleností $\|c_i - c_j\|$ a je-li více takových
114 prvků, pak se rozhoduje podle pořadí barvy c_i , popřípadě podle barvy c_j . Připomeňme, že
115 barvy jsou také uspořádány lexikograficky. Za předpokladu, že paleta neobsahuje duplicitu,
116 je funkce \min_{LEX} dobře definovaná. Algoritmus nijak nezávisí na uspořádání palety, což
117 je zřejmé z toho, že nikde nedochází k porovnávání indexů i a j .

118 Algoritmus skončí, protože množina E je konečná a při každém průchodu cyklem se
119 zmenší o alespoň jeden prvek, konkrétně o prvek (d, c_i, c_j) vybraný funkcí \min_{LEX} . Po
120 každém průchodu hlavním cyklem platí pro všechna $i \in I$ následující tři invarianty

$$\begin{aligned}
p(i) \text{ není definované} &\Leftrightarrow (E \cap (\mathbb{R} \times \{c_i\} \times C)) \neq \emptyset, \\
p(i) \text{ je definované} &\Leftrightarrow f(i) \text{ je definované}, \\
p(i) \text{ je definované} &\Rightarrow p(i) = 1 - p(f(i)).
\end{aligned}$$

121 Vzhledem k tomu, že na konci algoritmu je množina E prázdná, z prvního invariantu
122 plyne, že $p(i) \in \{0, 1\}$ pro všechna $i \in I$. Z druhého invariantu potom plyne, že také f je
123 řádně definované zobrazení, a ze třetího, že f obrací paritu.

124 Zbývá ukázat, že pro každé $i \in I$ je $\|c_i - c_{f(i)}\| = \min_{j \in I \setminus \{i\}} \|c_i - c_j\|$. Hlavní cyklus
125 algoritmu prochází dvojice barev (c_i, c_j) od nejpodobnějších dvojic po nejméně podobné
126 dvojice, a takto postupně předepisuje $f(i) := j$, popřípadě také $f(j) := i$. Potřebujeme
127 se tedy akorát přesvědčit, že jakmile je hodnota zobrazení f jednou definovaná pro určitý
128 index, už se v pozdější fázi algoritmu nepřepíše. K přepsání by mohlo dojít pouze operací
129 na řádku 7, 8 nebo 12. Složením prvního a druhého invariantu vidíme, že jakmile je $f(i)$
130 definované, je $E \cap (\mathbb{R} \times \{c_i\} \times C) = \emptyset$ a není tedy možné, aby funkce \min_{LEX} vrátila
131 prvek s barvou c_i na druhé pozici. Na řádcích 7 a 12 proto nemůže dojít k přepsání již
132 jednou definované hodnoty zobrazení f . Podle druhého invariantu platí, že jakmile je $f(j)$
133 definované, je $p(j) \neq \infty$, čili podmínka na řádku 4 není splněna. To znamená, že ani na 8.
134 řádku nemůže dojít k přepsání. \square

135 2. STEGANOGRAFIE POMOCÍ MATICOVÉHO VKLÁDÁNÍ

136 2.1. ÚVOD

137 Uvažujme stegosystém, který rozděluje prvky nosiče na tříprvkové bloky a do každého
138 bloku vkládá dva bity. Při standardním vkládání do nejnižšího bitu bychom u každého

139 bloku vložili bity zprávy např. do prvních dvou prvků a třetí prvek bychom nechali nedo-
 140 tčený. Jinou možností je rozložit bity zprávy do nejnižších bitů celého bloku takovým způ-
 141 sobem, aby příjemce z každého bloku stegoobjektu (y_1, y_2, y_3) extrahoval zprávu (z_1, z_2)
 142 jako $z_1 = \text{LSB}(y_1) \oplus \text{LSB}(y_2)$ a $z_2 = \text{LSB}(y_2) \oplus \text{LSB}(y_3)$. Vkládání v tomto případě pro-
 143 bíhá tak, že vyšetříme, která z následujících čtyř variant nastává a provedeme příslušnou
 144 úpravu:

$\text{LSB}(x_1) \oplus \text{LSB}(x_2)$	$\text{LSB}(x_2) \oplus \text{LSB}(x_3)$		y_1	y_2	y_3
z_1	z_2	\Rightarrow	x_1	x_2	x_3
$\neg z_1$	z_2	\Rightarrow	$\text{flip}(x_1)$	x_2	x_3
z_1	$\neg z_2$	\Rightarrow	x_1	x_2	$\text{flip}(x_3)$
$\neg z_1$	$\neg z_2$	\Rightarrow	x_1	$\text{flip}(x_2)$	x_3

146 Pro lepší náhled můžeme tuto tabulku porovnat s odpovídající tabulkou pro standardní
 147 vkládání do nejnižšího bitu:

$\text{LSB}(x_1)$	$\text{LSB}(x_2)$		y_1	y_2	y_3
z_1	z_2	\Rightarrow	x_1	x_2	x_3
$\neg z_1$	z_2	\Rightarrow	$\text{flip}(x_1)$	x_2	x_3
z_1	$\neg z_2$	\Rightarrow	x_1	$\text{flip}(x_2)$	x_3
$\neg z_1$	$\neg z_2$	\Rightarrow	$\text{flip}(x_1)$	$\text{flip}(x_2)$	x_3

149 Jak již víme, efektivita standardního vkládání do nejnižšího bitu je rovna 2 bez ohledu
 150 na délku nosiče či délku vložené zprávy. Při pohledu na poslední řádek v obou tabulkách
 151 vidíme, že nově představená metoda bude mít vyšší efektivitu vkládání. Předpokládáme-
 152 li, že všechny čtyři možnosti v první tabulce jsou stejně pravděpodobné, pak každý blok
 153 přispívá v průměru hodnotou $3/4$ k celkové distorzi vyvolané vkládáním. Počet vložených
 154 bitů je roven dvojnásobku počtu bloků. Efektivita vkládání je tedy $2b/(\frac{3}{4}b) = \frac{8}{3} \approx 2,667$,
 155 kde b je počet bloků nosiče. Vzhledem k tomu, že počet bloků nehraje roli, budeme jej
 156 napříště z našich úvah vynechávat. Nová metoda tedy vkládá o $\frac{2}{3}$ bitu víc na jednotku
 157 distorze oproti dřívější metodě. Ve skutečnosti se jedná jen o speciální případ metody
 158 známé jako *maticové vkládání*, která byla poprvé představena Crandalleem [3] v roce 1998.
 159 Tento název vychází z toho, že extrakci lze popsat pomocí maticového násobení

$$\begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} \text{LSB}(x_1) \\ \text{LSB}(x_2) \\ \text{LSB}(x_3) \end{pmatrix}.$$

160 Jak vidíme, je načase změnit značení, abychom mohli s objekty snáze pracovat po
 161 blocích a abychom se mohli oprostít od zobrazení LSB a flip. Vzhledem k tomu, že na
 162 nosiče a stegoobjekty přestaneme nahlížet jako na celky a budeme pracovat na nižší úrovni
 163 bloků, přestaneme se zároveň zatěžovat pojmem stegoklíče. Posloupnost hodnot spjatých
 164 s blokem nosiče a s blokem stegoobjektu budeme značit $\mathbf{x} = (x_1, \dots, x_n) \in \mathbb{F}_q^n$ a $\mathbf{y} \in \mathbb{F}_q^n$,
 165 kde n je velikost jednoho bloku. Blok zprávy budeme značit $\mathbf{z} \in \mathbb{F}_q^m$, kde m je počet
 166 q -árních symbolů vkládaných do jednoho bloku nosiče. Pod pojmem *spjatá hodnota* si
 167 pro $q = 2$ můžeme například představovat LSB příslušného prvku nosiče či stegoobjektu,
 168 nebo například paritu ve smyslu vkládání s optimálním přiřazením parity v paletových
 169 formátech. Zde budeme mít spjatou hodnotou na mysli zbytek po dělení prvku číslem q ,
 170 i když obecně může přicházet v úvahu i jiné přiřazení. Rozvláčnému výrazu „posloupnost
 171 hodnot spjatých s blokem“ se budeme dále vyhýbat, místo toho budeme pro jednoduchost
 172 hovořit přímo o \mathbf{x} jako o nosiči a o \mathbf{y} jako o stegoobjektu.

m	α	e
1	1,000	2,000
2	0,667	2,667
3	0,429	3,429
4	0,267	4,267
5	0,161	5,161
6	0,095	6,095
7	0,055	7,055
8	0,031	8,031
9	0,018	9,018

TABULKA 2.1: Relativní kapacita α a efektivita e Hammingových $[2^m - 1, 2^m - 1 - m]_2$ kódů.

173 Měli bychom zdůraznit, že zpráva se nyní skládá z q -árních symbolů, avšak kapacitu na-
174 dále měříme v bitech. Proto relativní kapacita nosiče $\alpha = (\log_2 q^{mb})/(nb) = m(\log_2 q)/n$,
175 kde b je počet bloků. Ani zde nehraje počet bloků roli, a můžeme jej tedy napříště vyne-
176 chávat.

177 2.2. MATICOVÉ VKLÁDÁNÍ POMOCÍ HAMMINGOVÝCH KÓDŮ

178 Nechť \mathbf{H} je paritní matice Hammingova $[\frac{q^m-1}{q-1}, \frac{q^m-1}{q-1} - m]_q$ kódu. Podle definice se jedná
179 o matici jejíž sloupce tvoří všechny nenulové vektory z \mathbb{F}_q^m , až na násobek. Pro začátek
180 uvažujme binární případ. Sloupce paritní matice je vhodné uspořádat lexikograficky, např.

$$\mathbf{H} = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

181 Extrakci zprávy ze stegoobjektu budeme provádět násobením paritní maticí zleva $\mathbf{z} = \mathbf{H}\mathbf{y}$.
182 Vkládání zprávy je malinko pracnější. Mějme nosič \mathbf{x} a zprávu \mathbf{z} . Spočítáme $\mathbf{H}\mathbf{x}$, tento
183 vektor se nazývá *syndrom* vektoru \mathbf{x} . Jestliže $\mathbf{H}\mathbf{x} \neq \mathbf{z}$, pak musíme určit změnu, kterou
184 je potřeba provést na \mathbf{x} , aby rovnost platila. V matici \mathbf{H} najdeme sloupec, který je roven
185 $\mathbf{z} - \mathbf{H}\mathbf{x}$. Označme jeho pořadí v matici jako j ; tento sloupec značíme \mathbf{H}_{*j} . Za stegoobjekt
186 zvolíme $\mathbf{y} = \mathbf{x} + \mathbf{e}_j$, kde \mathbf{e}_j je j -tý vektor kanonické báze prostoru \mathbb{F}_q^n . Snadno ověříme, že
187 $\mathbf{H}\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{H}\mathbf{e}_j = \mathbf{H}\mathbf{x} + \mathbf{H}_{*j} = \mathbf{z}$. Lexikografické uspořádání sloupců paritní matice
188 umožňuje okamžitě určit pořadové číslo sloupce j , protože sloupec je binární reprezentací
189 čísla j .

190 V podstatě stejný postup se používá při dekódování Hammingových kódů. Při dekódo-
191 vání je cílem získat nulový syndrom $\mathbf{H}\mathbf{y} = \mathbf{0}$, vektor \mathbf{e}_j je potom chybový vektor a $\mathbf{x} + \mathbf{e}_j$
192 je opravené slovo. Dekódování je tedy speciálním případem vkládání, a to vkládání nulové
193 zprávy.

194 Relativní kapacita stegosystému založeného na binárních Hammingových kódech je
195 $\alpha = m/n = m/(2^m - 1)$. Z popisu algoritmu vkládání vidíme, že je-li $\mathbf{H}\mathbf{x} \neq \mathbf{z}$, pak
196 distorze na blok je 1, jinak 0. Předpokládáme-li, že zprávy \mathbf{z} jsou voleny náhodně z \mathbb{F}_q^m
197 s rovnoměrným rozdělením, pak očekávaná distorze na blok je $(2^m - 1)/2^m = 1 - 2^{-m}$.
198 Efektivita vkládání je $e = m/(1 - 2^{-m})$ bitů na jednotku distorze.

199 Případ $m = 1$ vede na standardní vkládání do nejnižšího bitu a $m = 2$ odpovídá
200 úvodnímu příkladu. Tabulka 2.1 ukazuje relativní kapacitu a efektivitu binárních Ham-
201 mingových kódů pro 9 nejmenších hodnot m .

202 V praxi bychom vkládání pomocí Hammingových kódů realizovali následující způso-
 203 bem. Na vstupu dostaneme nosič a zprávu. Najdeme největší celé číslo m takové že

$$\frac{m}{2^m - 1} \geq \frac{\text{počet bitů zprávy}}{\text{počet prvků nosiče}}.$$

204 V nosiči vyhradíme místo, kam vložíme hodnotu m , a zbytek nosiče rozdělíme na bloky
 205 velikosti $2^m - 1$. Následně vkládáme pomocí Hammingova $[2^m - 1, 2^m - 1 - m]_2$ kódu.
 206 Tento postup používá stegosystém F5 pro vkládání do obrázků ve formátu JPEG.

207 Nyní přejdeme ke q -árním Hammingovým kódům. Jak už jsem zmínili v úvodu, prvky
 208 nosiče asociujeme s prvky tělesa \mathbb{F}_q pomocí operace $x \mapsto x \bmod q$. Jestliže q není prvočíslo,
 209 pak hodnotám $\{1, 2, \dots, q - 1\}$ přiřadíme prvky $\mathbb{F}_q \setminus \{0\}$ libovolným způsobem. Než bu-
 210 deme pokračovat, všimněme si, že i bez použití maticového vkládání lze tímto přístupem
 211 dosáhnout lepší efektivity vkládání než u klasického vkládání do nejnižšího bitu. Pro $q = 3$
 212 můžeme do každého prvku nosiče vložit jeden ternární symbol zprávy 0, 1, nebo 2 tak, že
 213 prvek necháme jak je, anebo jeho hodnotu zvýšíme či snížíme o 1, aby zbytek po dělení 3
 214 byl roven vkládanému symbolu zprávy. Do každého prvku tak vkládáme $\log_2 3 \approx 1,585$
 215 bitů a očekávaná distorze na každý prvek je $\frac{1}{3} \cdot 0^2 + \frac{1}{3} \cdot 1^2 + \frac{1}{3} \cdot (-1)^2 = 2/3$. Tímto docílíme
 216 efektivity vkládání $e = 3(\log_2 3)/2 \approx 2,377$. Pro vyšší hodnoty q už ovšem efektivita této
 217 metody klesá.

218 **Algoritmus 2.1** (vkládání pomocí Hammingova kódu).

vstup: nosič $\mathbf{x} \in \mathbb{F}_q^n$, zpráva $\mathbf{z} \in \mathbb{F}_q^m$,
 219 paritní matice \mathbf{H} Hammingova $[n, n - m]_q$ kódu
výstup: stegoobjekt $\mathbf{y} \in \mathbb{F}_q^n$ takový, že $\mathbf{H}\mathbf{y} = \mathbf{z}$
 1 **if** $\mathbf{H}\mathbf{x} = \mathbf{z}$ **then**
 2 **return** \mathbf{x}
 220 3 **else**
 4 najdi j -tý sloupec matice \mathbf{H} a $a \in \mathbb{F}_q$ takové, že $\mathbf{z} - \mathbf{H}\mathbf{x} = a\mathbf{H}_{*j}$
 5 **return** $\mathbf{x} + a\mathbf{e}_j$

221 *Důkaz.* $\mathbf{H}\mathbf{y} = \mathbf{H}\mathbf{x} + a\mathbf{H}\mathbf{e}_j = \mathbf{H}\mathbf{x} + a\mathbf{H}_{*j} = \mathbf{z}$. □

222 **Příklad 2.2.** Máme paritní matici Hammingova $[13, 10]_3$ kódu

$$\mathbf{H} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 2 & 2 & 2 \\ 1 & 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 \end{pmatrix}$$

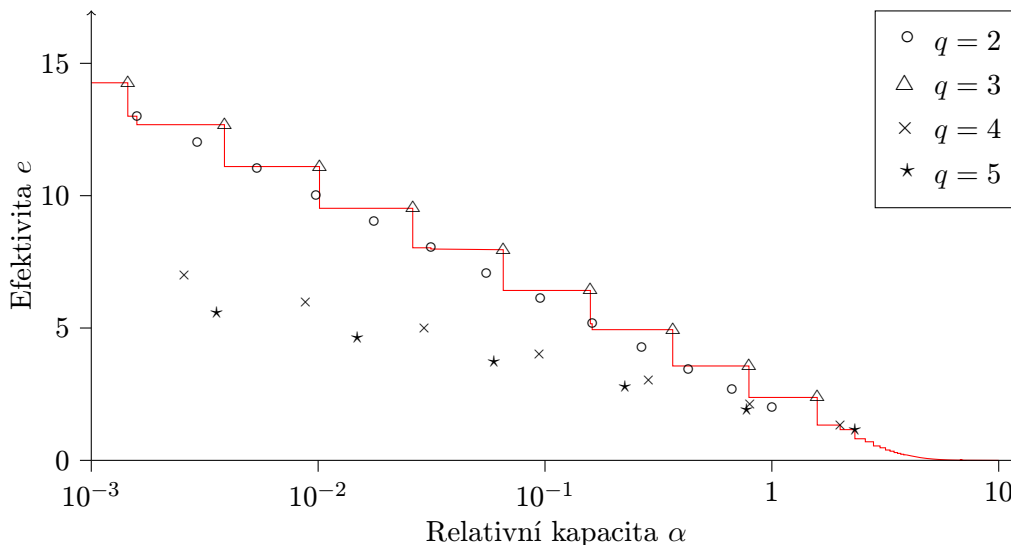
223 Do níže uvedeného bloku nosiče budeme vkládat zprávu $\mathbf{z} = (1, 2, 0)^T \in \mathbb{F}_3^3$. Složky vektoru
 224 \mathbf{x} získáme z prvků nosiče jako zbytek po dělení třemi.

225 Blok nosiče: 20 21 19 19 18 16 19 20 24 23 20 20 19
 $\mathbf{x} = (2 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 1 \quad 2 \quad 0 \quad 2 \quad 2 \quad 2 \quad 1)^T$

226 Spočteme $\mathbf{z} - \mathbf{H}\mathbf{x} = (1, 2, 0)^T - (2, 1, 1)^T = (2, 1, 2)^T$, což je rovno $2(1, 2, 1)^T = 2\mathbf{H}_{*12}$.
 227 Odtud $\mathbf{y} = \mathbf{x} + 2\mathbf{e}_{12}$. Blok nosiče upravíme takovým způsobem, aby zbytek po dělení
 228 jednotlivých prvků 3 dával \mathbf{y} . To lze v tomto případě docílit přičtením libovolné hodnoty
 229 z $2 + 3\mathbb{Z}$ na 12 pozici. Zvolíme pochopitelně tu, která je v absolutní hodnotě minimální,
 230 tj. -1 .

231 $\mathbf{y} = (2 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 1 \quad 2 \quad 0 \quad 2 \quad 2 \quad \mathbf{1} \quad 1)^T$
 Blok stegoobjektu: 20 21 19 19 18 16 19 20 24 23 20 **19** 19

232 Jedinou změnou velikosti 1 jsme vložili $3 \log_2 3 \approx 4,755$ bitů zprávy.



OBRÁZEK 2.1: Efektivita Hammingových $[\frac{q^m-1}{q-1}, \frac{q^m-1}{q-1} - m]_q$ kódů v závislosti na relativní kapacitě pro $q = 2, 3, 4$ a 5 .

233 Na závěr příkladu ještě poznamenejme, že vložení zprávy lze dosáhnout mnoha jinými
 234 způsoby. Například následující blok stegoobjektu nese tutéž zprávu, ale vznikl z bloku
 235 nosiče pomocí dvou změn velikosti 1.

236 **21 21 19 19 18 16 19 20 24 23 20 20 18**

237 Relativní kapacita vkládání pomocí q -árních Hammingových kódů je

$$\alpha = \frac{m \log_2 q}{(q^m - 1)/(q - 1)}.$$

238 Z algoritmu 2.1 vidíme, že je-li $\mathbf{H}\mathbf{x} \neq \mathbf{z}$, pak při vkládání zprávy \mathbf{z} dojde ke změně jediné
 239 hodnoty v bloku. Předpokládáme, že zprávy \mathbf{z} jsou voleny náhodně z \mathbb{F}_q^m s rovnoměrným
 240 rozdělením. Potom očekávaný počet změn na jeden blok je $(q^m - 1)/q^m = 1 - q^{-m}$. Otázkou
 241 zůstává, jaká je velikost změn. Pro q liché přicházejí v úvahu změny $\delta \in \Delta_q := \{\frac{-q+1}{2}, \dots,$
 242 $-1, 0, 1, \dots, \frac{q-1}{2}\}$ a pro q sudé $\delta \in \Delta_q := \{\frac{-q}{2} + 1, \dots, -1, 0, 1, \dots, \frac{q}{2}\}$. Všechny nenulové
 243 velikosti změn jsou stejně pravděpodobné. Očekávaný příspěvek jedné změny k distorzi je
 244 tudíž $(\sum_{\delta \in \Delta_q} \delta^2)/(q - 1)$. Efektivita vkládání je

$$e = \frac{(q - 1)m \log_2 q}{(1 - q^{-m}) \sum_{\delta \in \Delta_q} \delta^2}.$$

245 Na obrázku 2.1 máme graf efektivity Hammingových $[\frac{q^m-1}{q-1}, \frac{q^m-1}{q-1} - m]_q$ kódů v zá-
 246 vislosti na relativní kapacitě pro $q = 2, 3, 4, 5$ a několik nejnižších hodnot m . Červená
 247 čára vyznačuje nejvyšší možnou efektivitu dosažitelnou pomocí Hammingových kódů pro
 248 dané α . Ve většině případů se této maximální efektivity dosahuje ternárními kódy. Vý-
 249 jimečně může být výhodnější použít binární kód a pro $\alpha > \log_2 3$ v podstatě i víceární
 250 kódy. Pro $\alpha \geq 1$ se však ve skutečnosti už nejedná o maticové vkládání v pravém slova
 251 smyslu, jde totiž o vkládání pomocí $[1, 0]_q$ kódu.

252 Ačkoliv se ternární kódy jeví jako optimální, nesmíme zapomenout, že pro některé
 253 aplikace se hodí výhradně binární kódy. Příkladem může být vkládání při redukcí barevné

254 hloubky, kdy chceme zaokrouhlovat na nejbližší sudou či lichou hodnotu. Obdobně je tomu
 255 i v případě vkládání s optimálním přiřazením parity v paletových obrázcích. Konečně také
 256 při vkládání do obrázku JPEG pomocí dekrementace absolutních hodnot AC koeficientů,
 257 jako ve stegosystému F5.

258 Dále je třeba poznamenat, že neefektivita Hammingových kódů pro $q \geq 4$ je vázána
 259 na algoritmus vkládání 2.1. Tento algoritmus sice provádí nejvýše jednu změnu v každém
 260 bloku, avšak nezohledňuje velikost změny. V příkladu 2.2 jsme viděli, že vkládání lze re-
 261 alizovat více různými způsoby, např. také změnou dvou prvků v nosiči. Dvě malé změny
 262 mohou mít nižší příspěvek k distorzi než jedna velká. S algoritmem vkládání, který mini-
 263 malizuje nikoliv počet změn, ale celkový příspěvek změn k distorzi, bychom dospěli k vyšší
 264 efektivitě vkládání pro $q \geq 4$. Otázkou je, jaká by byla složitost takového algoritmu.

265 2.3. MATICOVÉ VKLÁDÁNÍ OBECNĚ

266 Na úvod připomeňme pár základních pojmů ze základního kurzu samoopravných kódů.
 267 Podprostor \mathcal{C} prostoru \mathbb{F}_q^n nazýváme *lineární kód* délky n . Dimenzi \mathcal{C} značíme k . Definu-
 268 jeme *Hammingovu váhu* $w(\mathbf{u})$ vektoru $\mathbf{u} \in \mathbb{F}_q^n$ jako počet nenulových složek v \mathbf{u} . *Ham-*
 269 *mingova vzdálenost* vektorů $\mathbf{u}, \mathbf{v} \in \mathbb{F}_q^n$ je $d_H(\mathbf{u}, \mathbf{v}) := w(\mathbf{v} - \mathbf{u})$ a Hammingova vzdálenost
 270 vektoru $\mathbf{u} \in \mathbb{F}_q^n$ od množiny $\mathcal{X} \subseteq \mathbb{F}_q^n$ je $d_H(\mathbf{u}, \mathcal{X}) = \min_{\mathbf{v} \in \mathcal{X}} w(\mathbf{v} - \mathbf{u})$. *Minimální vzdálenost*
 271 *neboli minimální váha* kódu \mathcal{C} je $\min_{\mathbf{c} \in \mathcal{C} \setminus \{\mathbf{0}\}} w(\mathbf{c})$. Lineární kód délky n a dimenze k nad
 272 \mathbb{F}_q s minimální vzdáleností d označujeme jako $[n, k, d]_q$ kód nebo zkráceně jen $[n, k]_q$ kód.
 273 Matici \mathbf{H} typu $(n - k) \times n$ nad tělesem \mathbb{F}_q s lineárně nezávislými řádky nazýváme *paritní*
 274 *maticí* kódu \mathcal{C} , jestliže

$$\mathcal{C} = \{ \mathbf{u} \in \mathbb{F}_q^n \mid \mathbf{H}\mathbf{u} = \mathbf{0} \}.$$

275 Prostor \mathbb{F}_q^n můžeme faktorizovat podle podprostoru \mathcal{C} . Definujeme *rozkladovou třídu*
 276 *příslušnou syndromu* $\mathbf{s} \in \mathbb{F}_q^{n-k}$

$$\mathcal{C}(\mathbf{s}) := \{ \mathbf{u} \in \mathbb{F}_q^n \mid \mathbf{H}\mathbf{u} = \mathbf{s} \}.$$

277 **Definice.** Nechť \mathbf{H} je paritní matice $[n, k]_q$ kódu \mathcal{C} a nechť $\mathbf{e} : \mathbb{F}_q^{n-k} \rightarrow \mathbb{F}_q^n$ je zobrazení
 278 takové, že pro všechna $\mathbf{s} \in \mathbb{F}_q^{n-k}$ je $\mathbf{H}\mathbf{e}(\mathbf{s}) = \mathbf{s}$ a $w(\mathbf{e}(\mathbf{s})) = \min_{\mathbf{u} \in \mathcal{C}(\mathbf{s})} w(\mathbf{u})$. Potom
 279 definujeme *maticovou extrakci* a *maticové vkládání*

$$\text{Ext}_{\mathbf{H}}(\mathbf{y}) := \mathbf{H}\mathbf{y} \quad \text{a} \quad \text{Emb}_{\mathbf{H}, \mathbf{e}}(\mathbf{x}, \mathbf{z}) := \mathbf{x} + \mathbf{e}(\mathbf{z} - \mathbf{H}\mathbf{x}),$$

280 kde $\mathbf{x}, \mathbf{y} \in \mathbb{F}_q^n$ a $\mathbf{z} \in \mathbb{F}_q^{n-k}$.

281 Vidíme, že $\text{Ext}_{\mathbf{H}}(\text{Emb}_{\mathbf{H}, \mathbf{e}}(\mathbf{x}, \mathbf{z})) = \mathbf{H}(\mathbf{x} + \mathbf{e}(\mathbf{z} - \mathbf{H}\mathbf{x})) = \mathbf{H}\mathbf{x} + \mathbf{z} - \mathbf{H}\mathbf{x} = \mathbf{z}$ pro každé
 282 $\mathbf{x} \in \mathbb{F}_q^n$ a $\mathbf{z} \in \mathbb{F}_q^{n-k}$. Místo $\text{Emb}_{\mathbf{H}, \mathbf{e}}$ budeme psát jen $\text{Emb}_{\mathbf{H}}$, protože nás zobrazení \mathbf{e} samo
 283 o sobě obvykle nezajímá. Podstatné je pouze to, že zobrazení \mathbf{e} zajišťuje, že počet změn
 284 vyvolaných vkládáním je minimalizován.

285 **Definice.** Nechť \mathcal{C} je $[n, k]_q$ kód. Zobrazení $D : \mathbb{F}_q^n \rightarrow \mathcal{C}$ nazýváme *minimum-distance*
 286 *dekodér* kódu \mathcal{C} , jestliže pro každé $\mathbf{u} \in \mathbb{F}_q^n$ platí $w(D(\mathbf{u}) - \mathbf{u}) = d_H(\mathbf{u}, \mathcal{C})$.

287 Jak jsme viděli v předchozí části, v případě Hammingových kódů je výpočet $\mathbf{e}(\mathbf{s})$
 288 jednoduchý. Spočívá toliko v nalezení toho sloupce paritní matice, který je násobkem \mathbf{s} .
 289 Z definice Hammingových kódů máme zaručeno, že takový sloupec v paritní matici exis-
 290 tuje. U jiných kódů už nemusí být výpočet $\mathbf{e}(\mathbf{s})$ tak jednoduchý, a v případě obecných
 291 kódů se dokonce jedná o NP-úplný problém [1]. Následující algoritmus nám dává návod,
 292 jak spočítat $\text{Emb}_{\mathbf{H}}$ pomocí minimum-distance dekodéru samoopravného kódu.

293 **Algoritmus 2.3** (maticové vkládání).

294 **vstup:** nosič $\mathbf{x} \in \mathbb{F}_q^n$, zpráva $\mathbf{z} \in \mathbb{F}_q^{n-k}$, paritní matice $\mathbf{H} [n, k]_q$ kódu \mathcal{C} ,
 294 minimum-distance dekodér D kódu \mathcal{C}

výstup: $\text{Emb}_{\mathbf{H}}(\mathbf{x}, \mathbf{z})$

1 zvol $\mathbf{u} \in \mathbb{F}_q^n$ takové, že $\mathbf{H}\mathbf{u} = \mathbf{z}$

2 **return** $D(\mathbf{x} - \mathbf{u}) + \mathbf{u}$

296 *Důkaz.* Abychom dokázali správnost algoritmu, stačí ověřit, že za $\mathbf{e}(\mathbf{z} - \mathbf{H}\mathbf{x})$ v definici
 297 maticového vkládání můžeme zvolit $D(\mathbf{x} - \mathbf{u}) - (\mathbf{x} - \mathbf{u})$. Prvním požadavkem je, aby
 298 $\mathbf{H}\mathbf{e}(\mathbf{z} - \mathbf{H}\mathbf{x}) = \mathbf{z} - \mathbf{H}\mathbf{x}$. To je zřejmě splněno

$$\mathbf{H}(D(\mathbf{x} - \mathbf{u}) - (\mathbf{x} - \mathbf{u})) = \mathbf{0} - \mathbf{H}\mathbf{x} + \mathbf{z}.$$

299 Druhým požadavkem je, aby $w(\mathbf{e}(\mathbf{z} - \mathbf{H}\mathbf{x})) = \min_{\mathbf{u} \in \mathcal{C}(\mathbf{z} - \mathbf{H}\mathbf{x})} w(\mathbf{u})$, a skutečně

$$w(D(\mathbf{x} - \mathbf{u}) - (\mathbf{x} - \mathbf{u})) = d_{\mathbf{H}}(\mathbf{x} - \mathbf{u}, \mathcal{C}) = \min_{\mathbf{c} \in \mathcal{C}} w(\mathbf{c} - (\mathbf{x} - \mathbf{u})) = \min_{\mathbf{v} \in \mathcal{C} + \mathbf{u} - \mathbf{x}} w(\mathbf{v}),$$

300 kde množina $\mathcal{C} + \mathbf{u} - \mathbf{x} = \mathcal{C}(\mathbf{H}(\mathbf{u} - \mathbf{x})) = \mathcal{C}(\mathbf{z} - \mathbf{H}\mathbf{x})$. □

301 K algoritmu maticového vkládání ještě poznamenejme, že je-li paritní matice v syste-
 302 matickém tvaru, tj. $\mathbf{H} = (\mathbf{I}_{n-k} \mid \mathbf{H}')$, pak první krok algoritmu je triviální. Za \mathbf{u} lze totiž
 303 zvolit vektor \mathbf{z} doplněný k nulami $\mathbf{u} = (\mathbf{z}^T \mid \mathbf{0}_k^T)^T$.

304 Budeme-li chtít v praxi používat maticové vkládání pro obecné samoopravné kódy,
 305 pak budeme muset slevit z požadavku minimality $w(\mathbf{e}(\mathbf{s}))$, abychom docílili lepší časové
 306 efektivity. To lze realizovat jednoduše tím, že minimum-distance dekodér v algoritmu 2.3
 307 nahradíme jiným, časově efektivnějším, dekodérem.

308 **Definice.** Pro každý $[n, k]_q$ kód \mathcal{C} definujeme *pokrývací poloměr kódu* \mathcal{C}

$$r_c := \max_{\mathbf{u} \in \mathbb{F}_q^n} d_{\mathbf{H}}(\mathbf{u}, \mathcal{C})$$

309 a *průměrnou (Hammingovu) vzdálenost od* \mathcal{C}

$$r_a := \frac{1}{q^n} \sum_{\mathbf{u} \in \mathbb{F}_q^n} d_{\mathbf{H}}(\mathbf{u}, \mathcal{C}).$$

310 **Věta 2.4** (o maticovém vkládání). *Nechť \mathcal{C} je $[n, k]_q$ kód s paritní maticí \mathbf{H} , pokrývacím*
 311 *poloměrem r_c a průměrnou vzdáleností od kódu r_a . Potom*

$$\max_{\mathbf{x}, \mathbf{z}} d_{\mathbf{H}}(\mathbf{x}, \text{Emb}_{\mathbf{H}}(\mathbf{x}, \mathbf{z})) = r_c \quad a \quad \mathbb{E}_{\mathbf{x}, \mathbf{z}} [d_{\mathbf{H}}(\mathbf{x}, \text{Emb}_{\mathbf{H}}(\mathbf{x}, \mathbf{z}))] = r_a,$$

312 kde $\mathbf{x} \in \mathbb{F}_q^n$ a $\mathbf{z} \in \mathbb{F}_q^{n-k}$.

313 *Důkaz.* Nechť $\mathbf{s} \in \mathbb{F}_q^{n-k}$, pak pro každé $\mathbf{u} \in \mathcal{C}(\mathbf{s})$ platí $d_{\mathbf{H}}(\mathbf{u}, \mathcal{C}) = w(\mathbf{e}(\mathbf{s}))$, neboť

$$d_{\mathbf{H}}(\mathbf{u}, \mathcal{C}) = \min_{\mathbf{c} \in \mathcal{C}} w(\mathbf{u} - \mathbf{c}) = \min_{\mathbf{v} \in \mathcal{C} + \mathbf{u}} w(\mathbf{v}) = \min_{\mathbf{v} \in \mathcal{C}(\mathbf{s})} w(\mathbf{v}) = w(\mathbf{e}(\mathbf{s})).$$

První část tvrzení je dokázána následujícími rovnostmi. Nechť $\mathbf{x} \in \mathbb{F}_q^n$, potom

$$\begin{aligned} \max_{\mathbf{z} \in \mathbb{F}_q^{n-k}} d_{\mathbf{H}}(\mathbf{x}, \text{Emb}_{\mathbf{H}}(\mathbf{x}, \mathbf{z})) &= \max_{\mathbf{z} \in \mathbb{F}_q^{n-k}} w(\mathbf{e}(\mathbf{z} - \mathbf{H}\mathbf{x})) = \\ &= \max_{\mathbf{s} \in \mathbb{F}_q^{n-k}} w(\mathbf{e}(\mathbf{s})) = \max_{\substack{\mathbf{s} \in \mathbb{F}_q^{n-k} \\ \mathbf{u} \in \mathcal{C}(\mathbf{s})}} d_{\mathbf{H}}(\mathbf{u}, \mathcal{C}) = \max_{\mathbf{u} \in \mathbb{F}_q^n} d_{\mathbf{H}}(\mathbf{u}, \mathcal{C}) = r_c. \end{aligned}$$

314 Předposlední rovnost plyne z toho, že \mathbb{F}_q^n je disjunktním sjednocením $\mathcal{C}(\mathbf{s})$ přes všechna
 315 $\mathbf{s} \in \mathbb{F}_q^{n-k}$.

V důkazu druhé části předpokládáme, že zpráva \mathbf{z} je volena náhodně z \mathbb{F}_q^{n-k} s rovno-
 měrným rozdělením.

$$\begin{aligned} \mathbb{E}_{\mathbf{x}, \mathbf{z}} [d_H(\mathbf{x}, \text{Emb}_H(\mathbf{x}, \mathbf{z}))] &= \mathbb{E}_{\mathbf{s}} [w(\mathbf{e}(\mathbf{s}))] = \frac{1}{q^{n-k}} \sum_{\mathbf{s} \in \mathbb{F}_q^{n-k}} w(\mathbf{e}(\mathbf{s})) = \\ &= \frac{1}{q^n} \sum_{\mathbf{s} \in \mathbb{F}_q^{n-k}} q^k w(\mathbf{e}(\mathbf{s})) = \frac{1}{q^n} \sum_{\mathbf{s} \in \mathbb{F}_q^{n-k}} \sum_{\mathbf{u} \in \mathcal{C}(\mathbf{s})} d_H(\mathbf{u}, \mathcal{C}) = \frac{1}{q^n} \sum_{\mathbf{u} \in \mathbb{F}_q^n} d_H(\mathbf{u}, \mathcal{C}) = r_a. \end{aligned}$$

316 Předposlední rovnost opět plyne z toho, že \mathbb{F}_q^n je disjunktním sjednocením $\mathcal{C}(\mathbf{s})$ přes
 317 všechna $\mathbf{s} \in \mathbb{F}_q^{n-k}$. \square

318 V případě těles \mathbb{F}_2 a \mathbb{F}_3 je distorze prakticky rovna počtu změn vyvolaných vkládá-
 319 ním, protože téměř všechny změny lze uskutečnit přičtením nebo odečtením hodnoty 1
 320 od příslušného prvku nosiče. Výjimka může nastat v případě, kdy pracujeme s kódem
 321 nad tělesem \mathbb{F}_3 a prvky nosiče mají omezený obor hodnot, např. $\{0, 1, \dots, 255\}$. Problém
 322 nastává pro krajní hodnoty takového intervalu. Vkládáme-li hodnotu 2 do prvku s hod-
 323 notou 0 nebo hodnotu 1 do prvku s hodnotou 255 musíme v prvním případě provést
 324 změnu $+2$ a ve druhém případě změnu -2 . Zanedbáme-li tyto speciální případy, můžeme
 325 vyslovit následující.

326 **Důsledek 2.5.** *Jestliže $q \in \{2, 3\}$, pak efektivita maticového vkládání pro $[n, k]_q$ kód \mathcal{C} je*
 327 *$e = (n - k)(\log_2 q)/r_a$, kde r_a je průměrná vzdálenost od kódu.*

328 V případě těles \mathbb{F}_q , $q \geq 4$, je situace o něco složitější. Připomeňme, že pro q liché přichá-
 329 zejí v úvahu změny $\delta \in \Delta_q := \{-\frac{q+1}{2}, \dots, -1, 0, 1, \dots, \frac{q-1}{2}\}$ a pro q sudé $\delta \in \Delta_q := \{-\frac{q}{2}+1,$
 330 $\dots, -1, 0, 1, \dots, \frac{q}{2}\}$. Předpokládáme, že každá nenulová změna je stejně pravděpodobná.
 331 Distorze potom vychází $r_a(q-1)^{-1} \sum_{\delta \in \Delta_q} \delta^2$, kde r_a je průměrná vzdálenost od přísluš-
 332 ného kódu.

333 Jak již víme, maximální možný počet změn vyvolaných maticovým vkládáním je roven
 334 pokrývacímu poloměru r_c . Distorze je proto shora omezena hodnotou $r_c(q-1)^{-1} \sum_{\delta \in \Delta_q} \delta^2$,
 335 což nám naopak dává následující spodní mez na efektivitu vkládání.

336 **Definice.** Pro $[n, k]_q$ kód s pokrývacím poloměrem r_c definujeme *spodní efektivitu* vklá-
 337 dání

$$\underline{e} := \frac{n\alpha(q-1)}{r_c \sum_{\delta \in \Delta_q} \delta^2},$$

338 kde $\alpha = (1 - \frac{k}{n}) \log_2 q$.

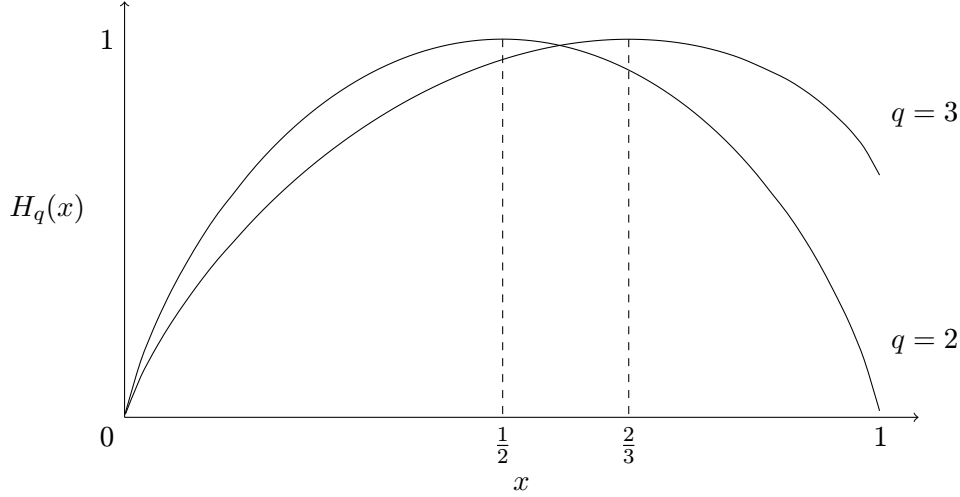
339 2.4. SPODNÍ EFEKTIVITA VKLÁDÁNÍ A VELIKOST TĚLESA

340 Cílem této části je nahlédnout, jak velikost tělesa ovlivňuje efektivitu vkládání, a pokusit
 341 se odhalit optimální velikost tělesa. Při cestě k tomuto cíli budeme mimo jiné potřebovat
 342 zobecněnou definici entropické funkce.

343 **Definice.** Pro každé celé číslo $q \geq 2$ definujeme q -ární entropickou funkci $H_q : [0, 1] \rightarrow \mathbb{R}$
 344 předpisem

$$H_q(x) := -x \log_q x - (1-x) \log_q(1-x) + x \log_q(q-1), \quad x \in (0, 1),$$

345 a v krajních bodech intervalu definujeme spojitě $H_q(0) = 0$ a $H_q(1) = \log_q(q-1)$.



OBRÁZEK 2.2: q -ární entropická funkce H_q pro $q = 2$ a 3 .

346 Na obrázku 2.2 máme graf q -ární entropické funkce pro $q = 2$ a $q = 3$.

347 Pro $\mathbf{v} \in \mathbb{F}_q^n$ a $r \geq 0$, definujeme kouli se středem \mathbf{v} a poloměrem r jako $\mathcal{B}(\mathbf{v}, r) := \{\mathbf{u} \in$
 348 $\mathbb{F}_q^n \mid d_H(\mathbf{u}, \mathbf{v}) \leq r\}$. Počet prvků libovolné koule v \mathbb{F}_q^n o poloměru r značíme $V_q(n, r)$. Jak
 349 známo $V_q(n, r) = \sum_{i=0}^r \binom{n}{i} (q-1)^i$

350 **Lemma 2.6.** *Nechť $0 \leq \frac{r}{n} \leq 1 - q^{-1}$, potom $n H_q(\frac{r}{n}) \geq \log_q V_q(n, r)$.*

351 *Důkaz.* Na úvod si všimněme toho, že nerovnost $\frac{r}{n} \leq 1 - q^{-1}$ uvedenou v předpokladu
 352 lemmatu můžeme přepsat $1 - (1 - \frac{r}{n})^{-1} \geq 1 - q$ a dále jako $\frac{r}{(n-r)(q-1)} \leq 1$. Pro $r = 0$
 353 lemma zřejmě platí, dále tedy předpokládejme $r > 0$.

Začneme tím, že výraz $q^{nH_q(r/n)}$ rozepíšeme podle definice entropické funkce.

$$q^{nH_q(r/n)} = \left(\frac{r}{n}\right)^{-r} \left(1 - \frac{r}{n}\right)^{-(n-r)} (q-1)^r = \frac{n^n}{r^r (n-r)^{n-r}} (q-1)^r$$

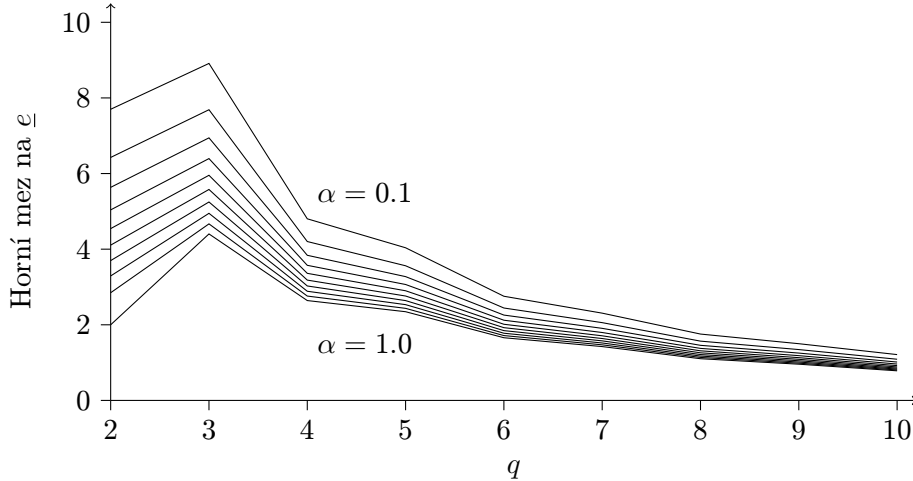
Dále vyjádříme n^n pomocí binomického rozvoje a získáme spodní odhad.

$$\begin{aligned} n^n &= (r + (n-r))^n = \sum_{i=0}^n \binom{n}{i} r^i (n-r)^{n-i} \frac{(q-1)^i}{(q-1)^i} \\ &\geq \sum_{i=0}^r \binom{n}{i} \left(\frac{r}{(n-r)(q-1)}\right)^i (n-r)^n (q-1)^i \\ &\geq \sum_{i=0}^r \binom{n}{i} \left(\frac{r}{(n-r)(q-1)}\right)^r (n-r)^n (q-1)^i \end{aligned}$$

354 V posledním kroku jsme využili předpokladu, že $\frac{r}{(n-r)(q-1)} \leq 1$. Po dosazení nerovnosti
 355 za n^n vidíme, že platí $q^{nH_q(r/n)} \geq \sum_{i=0}^r \binom{n}{i} (q-1)^i$, což jsme měli dokázat. \square

356 Z derivace entropické funkce $H'_q(x) = \log_q((1-q)(1-x^{-1}))$ lze snadno ověřit, že H_q
 357 je na intervalu $(0, 1 - q^{-1})$ rostoucí. Funkce H_q je navíc na $[0, 1 - q^{-1}]$ spojitá a její obor
 358 hodnot je tedy $[0, 1]$. Můžeme proto definovat inverzní funkci $H_q^{-1} : [0, 1] \rightarrow [0, 1 - q^{-1}]$.

359 **Věta 2.7.** *Pro každý $[n, k]_q$ kód s pokrývacím poloměrem r_c platí $r_c \geq n H_q^{-1}(\alpha / \log_2 q)$,
 360 kde $\alpha = (1 - \frac{k}{n}) \log_2 q$.*



OBRÁZEK 2.3: Horní mez $\bar{e}(q, \alpha)$ na spodní efektivitu vkládání v závislosti na q pro $\alpha = 0,1, 0,2, \dots, 1,0$.

361 *Důkaz.* Jestliže $\frac{r_c}{n} \geq 1 - q^{-1}$ pak tvrzení věty platí triviálně. Dále tedy předpokládejme,
362 že $\frac{r_c}{n} < 1 - q^{-1}$. Pokrývací poloměr kódu \mathcal{C} můžeme také chápat jako nejmenší r_c takové,
363 že $\bigcup_{\mathbf{c} \in \mathcal{C}} \mathcal{B}(\mathbf{c}, r_c) = \mathbb{F}_q^n$. Jinými slovy pro každý $[n, k]_q$ kód s pokrývacím poloměrem r_c platí
364 $q^n \leq q^k V_q(n, r_c)$. Podle lemmatu 2.6 dostáváme horní odhad na délku zprávy $n - k \leq$
365 $\log_q V_q(n, r_c) \leq n H_q(\frac{r_c}{n})$. Vzhledem k tomu, že H_q je na intervalu $[0, 1 - q^{-1}]$ rostoucí,
366 máme $H_q^{-1}(1 - \frac{k}{n}) \leq \frac{r_c}{n}$. \square

367 Z předchozí věty plyne horní mez na spodní efektivitu maticového vkládání v závislosti
368 na relativní kapacitě α

$$\bar{e}(q, \alpha) := \frac{\alpha(q-1)}{H_q^{-1}(\alpha/\log_2 q) \sum_{\delta \in \Delta_q} \delta^2} \geq \underline{e}. \quad (2.1)$$

369 Na obrázku 2.3 máme grafy této horní meze v závislosti na q pro několik různých hodnot α .
370 Grafy naznačují, že chceme-li pro pevně zvolené α dosáhnout nejvyšší možné efektivitu
371 maticového vkládání, měli bychom použít nějaký ternární kód. Tato úvaha ovšem stojí na
372 několika nepotvrzených předpokladech. Především nemáme dokázáno, že horní mez (2.1) je
373 skutečně dosažitelná, jinými slovy, že pro každé α skutečně existují ternární kódy, jejichž
374 pokrývací poloměr je libovolně blízko k mezi uvedené ve větě 2.7. V případě binárních
375 kódů tomu tak je. Přesněji řečeno, je známo, že pro každé α a $\epsilon > 0$ existuje $[n, k]_2$ kód
376 s pokrývacím poloměrem nejvýše $n H_2^{-1}(\alpha)$ takový, že $|1 - \frac{k}{n} - \alpha| \leq \epsilon$. To plyne z následující
377 věty, kde za ϱ zvolíme $H_2^{-1}(\alpha)$.

378 **Věta 2.8** ([2], Theorem 12.3.5). *Pro každé $\varrho \in [0, \frac{1}{2})$ existuje posloupnost celých čísel*
379 *$\{k_n\}_{n=1}^\infty$ taková, že*

$$k_n/n \leq 1 - H_2(\varrho) + O(n^{-1} \log n)$$

380 *a podíl všech $[n, k_n]_2$ kódů s pokrývacím poloměrem nejvýše ϱn konverguje k 1 pro $n \rightarrow \infty$.*

381 Dále je třeba mít na paměti, že (2.1) není horní mezí na efektivitu, ale na spodní efek-
382 tivitu. Efektivita vkládání je přitom pro daný kód vždy vyšší než jeho spodní efektivita.
383 Rozdíl mezi těmito dvěma veličinami spočívá v tom, že místo očekávaného počtu změn r_a
384 vyvolaných vkládáním počítáme u spodní efektivitě s maximálním počtem změn r_c . Aby

385 tedy pro nějaký kód byla spodní efektivita kódu dobrou aproximací jeho skutečné efek-
 386 tivity, je třeba, aby hodnoty r_c a r_a tohoto kódu měly k sobě blízko. Následující věta
 387 potvrzuje, že alespoň v případě „většiny“ binárních kódů tomu tak skutečně je.

388 **Věta 2.9** ([7], Theorem 4). *Nechť $0 < \alpha < 1$ a $\varepsilon > 0$. Podíl všech $[n, (1 - \alpha)n]_2$ kódů, pro
 389 něž platí $|r_c - r_a|/n \leq \varepsilon$, konverguje k 1 pro $n \rightarrow \infty$.*

390 Konečně nesmíme zapomenout, že celá naše úvaha stojí na předpokladu, že statistickou
 391 detekovatelnost vkládání lze měřit pomocí distorze, což je ovšem otevřená otázka.

392 2.5. PERFEKTNÍ KÓDY A HORNÍ MEZ NA EFEKTIVITU VKLÁDÁNÍ

393 **Tvrzení 2.10.** *Nechť \mathcal{C} je $[n, k]_q$ kód s průměrnou vzdáleností r_a a nechť r je největší celé
 394 číslo takové, že $V_q(n, r) \leq q^{n-k}$. Potom*

$$r_a \geq q^{k-n} \sum_{i=1}^r i(q-1)^i \binom{n}{i},$$

395 *přičemž rovnost nastává právě tehdy, když \mathcal{C} je perfektní kód.*

396 *Důkaz.* Označme \mathcal{D} multimnožinu $\{w(\mathbf{c} - \mathbf{u}) \mid \mathbf{u} \in \mathbb{F}_q^n, \mathbf{c} \in \mathcal{C}\}$. Každá hodnota $i \in$
 397 $\{0, 1, \dots, n\}$ má v \mathcal{D} četnost $q^k(q-1)^i \binom{n}{i}$. Naším cílem je získat spodní odhad výrazu
 398 $\sum_{\mathbf{u} \in \mathbb{F}_q^n} \min_{\mathbf{c} \in \mathcal{C}} w(\mathbf{c} - \mathbf{u})$. Tento výraz vyjadřuje součet určité q^n -prvkové podmnožiny
 399 multimnožiny \mathcal{D} , a můžeme ho proto zespolu odhadnout součtem q^n nejmenších hodnot
 400 z \mathcal{D} . Podle předpokladu je $\sum_{i=0}^r q^k(q-1)^i \binom{n}{i} \leq q^n$, a následující spodní odhad je tedy
 401 součtem nejvýše q^n nejmenších hodnot z \mathcal{D}

$$\sum_{\mathbf{u} \in \mathbb{F}_q^n} \min_{\mathbf{c} \in \mathcal{C}} w(\mathbf{c} - \mathbf{u}) \geq \sum_{i=0}^r i \cdot q^k(q-1)^i \binom{n}{i}.$$

402 Jestliže \mathcal{C} je perfektní kód s minimální vahou d , pak $r = (d-1)/2$ a \mathbb{F}_q^n je disjunktním
 403 sjednocením $\mathcal{B}_q(\mathbf{c}, r)$ přes všechna $\mathbf{c} \in \mathcal{C}$. V takovém případě zřejmě nastává rovnost.

404 Naopak, nastává-li rovnost, pak $q^k V_q(n, r) = q^n$ a v součtu $\sum_{\mathbf{u} \in \mathbb{F}_q^n} \min_{\mathbf{c} \in \mathcal{C}} w(\mathbf{c} - \mathbf{u})$
 405 sčítáme právě q^n nejmenších hodnot z \mathcal{D} . Každá z těchto hodnot je nejvýše r , a proto
 406 každé $\mathbf{u} \in \mathbb{F}_q^n$ leží ve sjednocení $\bigcup_{\mathbf{c} \in \mathcal{C}} \mathcal{B}_q(\mathbf{c}, r)$. Jelikož $q^k V_q(n, r) = q^n$, musí se jednat
 407 o disjunktní sjednocení, tím pádem o perfektní kód. \square

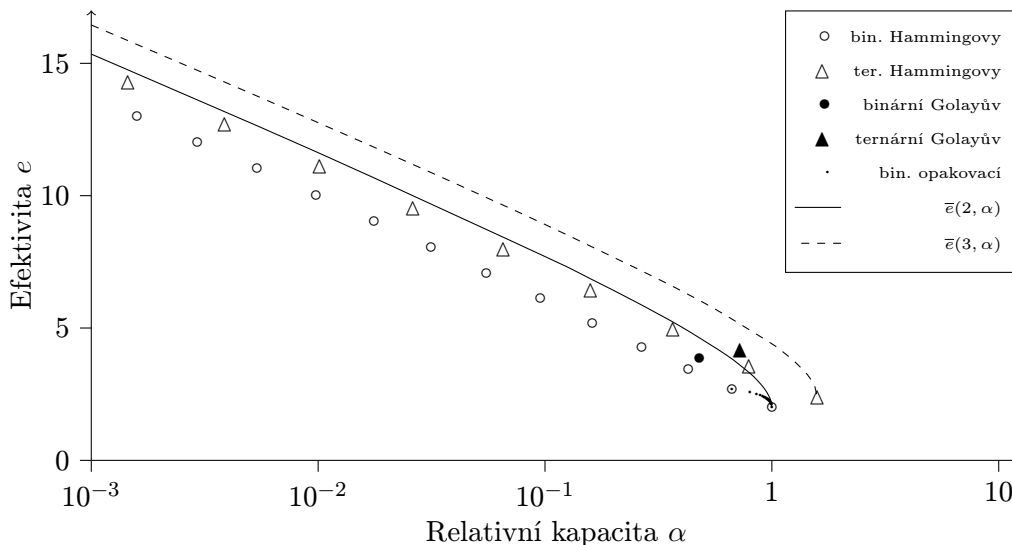
408 Připomeňme, že mezi perfektní kódy patří všechny úplné kódy, jednobodové kódy a
 409 binární opakovací kódy liché délky. Tyto jsou známy jako triviální perfektní kódy. Každý
 410 netriviální perfektní kód má parametry Hammingova kódu, Golayova $[23, 12, 7]_2$ kódu,
 411 anebo Golayova $[11, 6, 5]_3$ kódu.

412 Spojením předchozího tvrzení s důsledkem 2.5 získáme následující.

413 **Důsledek 2.11.** *Nechť \mathcal{C} je $[n, k]_q$ kód, kde $q \in \{2, 3\}$, a nechť r je největší celé číslo
 414 takové, že $V_q(n, r) \leq q^{n-k}$. Potom efektivita maticového vkládání pro kód \mathcal{C} je*

$$e \leq \frac{q^{n-k}(n-k) \log_2 q}{\sum_{i=1}^r i(q-1)^i \binom{n}{i}},$$

415 *přičemž rovnost nastává právě tehdy, když \mathcal{C} je perfektní kód.*



OBRÁZEK 2.4: Efektivita binárních a ternárních perfektních kódů v závislosti na relativní kapacitě ve srovnání s horní mezí na spodní efektivitu pro $q = 2$ a 3 .

416 Tento důsledek nám říká, že zvolíme-li pevně n , α a q , a budeme vybírat mezi všemi
 417 $[n, n(1 - \alpha/\log_2 q)]_q$ kódy, pak nejvyšší efektivitu vkládání dosáhneme volbou perfektního
 418 kódu (existuje-li perfektní kód s těmito parametry). Toto ovšem neznamená, že by
 419 perfektní kódy byly nejlepší volbou za každých podmínek. Na obrázku 2.4 máme graf efek-
 420 tivity různých binárních a ternárních perfektních kódů v závislosti na relativní kapacitě.
 421 Vidíme, že perfektní kódy obecně nedosahují příslušnou horní mezí na spodní efektivitu
 422 vkládání. Z předchozí části přitom víme, že zvolíme-li pevně pouze α a q , a připustíme,
 423 aby n bylo libovolně velké, pak alespoň pro $q = 2$ existují kódy s těmito parametry, kte-
 424 rými se lze přiblížit libovolně blízko k mezi $\bar{e}(2, \alpha)$. Nicméně perfektní kódy nemají svou
 425 efektivitou daleko k příslušné mezi. Odstup od horní meze na spodní efektivitu je do výše
 426 1,7.

427 **Příklad 2.12.** Díky důsledku 2.11 můžeme jednoduše spočítat efektivitu vkládání pro
 428 Golayovy kódy. Stačí za r dosadit $(d - 1)/2$, kde d je minimální váha kódu. Pro binární
 429 Golayův kód máme $\alpha = 11/23$ a $e = 11 \cdot 2^{11} / \sum_{i=1}^3 i \binom{23}{i} = 11264/2921 \approx 3,856$. Pro
 430 ternární Golayův kód máme $\alpha = 5(\log_2 3)/11 \approx 0,720$ a $e = 5 \cdot 3^5 (\log_2 3) / \sum_{i=1}^2 i \cdot 2^i \binom{11}{i} =$
 431 $1215(\log_2 3)/462 \approx 4,168$.

432 **Cvičení 2.13.** Ukažte, že efektivita binárního opakovacího kódu liché délky n je

$$\frac{2(n-1)}{n \left(1 - 2^{1-n} \binom{n-1}{(n-1)/2} \right)}.$$

433 3. SOUČTOVĚ A ROZDÍLOVĚ POKRÝVACÍ MNOŽINY

434 V této části představíme metodu vkládání, která zobecňuje maticové vkládání ternárními
 435 kódy. Začneme jednoduchým příkladem schématu, které rozděljuje nosič na dvouprvkové

$x_1 + 2x_2$	zpráva z			
	0	1	2	3
0	(0, 0)	(+1, 0)	(0, ±1)	(-1, 0)
1	(-1, 0)	(0, 0)	(+1, 0)	(0, ±1)
2	(0, ±1)	(-1, 0)	(0, 0)	(+1, 0)
3	(+1, 0)	(0, ±1)	(-1, 0)	(0, 0)

TABULKA 3.1: Vkládání pomocí $(2, 1, \mathbb{Z}_4)$ -SDCS.

bloky (x_1, x_2) a do každého bloku vkládá kvaternární symbol $z \in \mathbb{Z}_4$. Extrakce je definována jako $\text{Ext}(y_1, y_2) = (y_1 + 2y_2) \bmod 4$. Vkládání probíhá tak, že nejdříve provedeme extrakci na bloku nosiče a ověříme, zda je výsledek roven z . Jestliže není, změníme blok nosiče tak, jak je uvedeno v tabulce 3.1. Například jestliže $\text{Ext}(x_1, x_2) = (x_1 + 2x_2) \bmod 4 = 3$, ale my chceme do bloku vložit symbol 2, pak stačí snížit x_1 o 1, čili $(y_1, y_2) = (x_1, x_2) + (-1, 0)$. Očekávaný příspěvek k celkové distorzi je $\frac{3}{4}$ pro každý blok. V každém bloku máme 2 bity zprávy, čili efektivita je $e = 2/\frac{3}{4} = \frac{8}{3} \approx 2,667$ a relativní kapacita je $\alpha = 2/2 = 1$. To je zatím nejlepší schéma s takto vysokou kapacitou, které jsme dosud viděli. Nyní toto schéma zobecníme.

Definice. Nechť n a r jsou přirozená čísla, $(G, +)$ je konečná abelovská grupa a $\mathcal{A} = \{a_1, \dots, a_n\}$ je posloupnost po dvou různých hodnot z G . Jestliže pro každé $g \in G$ existují $s_1, \dots, s_n \in \{-1, 0, 1\}$ takové, že

$$\sum_{i=1}^n |s_i| \leq r \quad \text{a} \quad \sum_{i=1}^n s_i a_i = g,$$

pak říkáme, že \mathcal{A} je *součtově a rozdílově pokrývací množina* (sum and difference covering set, SDCS) s parametry (n, r, G) . Pro každé $g \in G$ definujeme *SDCS váhu* prvku g

$$w_{\mathcal{A}}(g) := \min \left\{ \sum_{i=1}^n |s_i| \mid s_1, \dots, s_n \in \{-1, 0, 1\}, \sum_{i=1}^n s_i a_i = g \right\}.$$

Úvodní příklad můžeme popsat jako SDCS $\{1, 2\}$ s parametry $(2, 1, \mathbb{Z}_4)$.

Všimněme si, že z každého $[n, k]_3$ kódu s pokrývacím poměrem r_c lze sestavit SDCS s parametry $(n, r_c, \mathbb{F}_3^{n-k})$ jednoduše tak, že za \mathcal{A} zvolíme sloupce jeho paritní matice.

Definice. Nechť $\mathcal{A} = (a_1, \dots, a_n)$ je SDCS a $(y_1, \dots, y_n) \in \mathbb{Z}^n$ je blok stegoobjektu. Definujeme *SDCS extrakci* z bloku (y_1, \dots, y_n) jako

$$\text{Ext}_{\mathcal{A}}(y_1, \dots, y_n) := \sum_{i=1}^n y_i a_i.$$

Algoritmus 3.1 (SDCS vkládání).

vstup: SDCS $\mathcal{A} = (a_1, \dots, a_n) \in G^n$, blok nosiče $(x_1, \dots, x_n) \in \mathbb{Z}^n$, zpráva $z \in G$

výstup: blok stegoobjektu $(y_1, \dots, y_n) \in \mathbb{Z}^n$ takový, že $\text{Ext}_{\mathcal{A}}(y_1, \dots, y_n) = z$

1 $g := z - \sum_{i=1}^n x_i a_i$

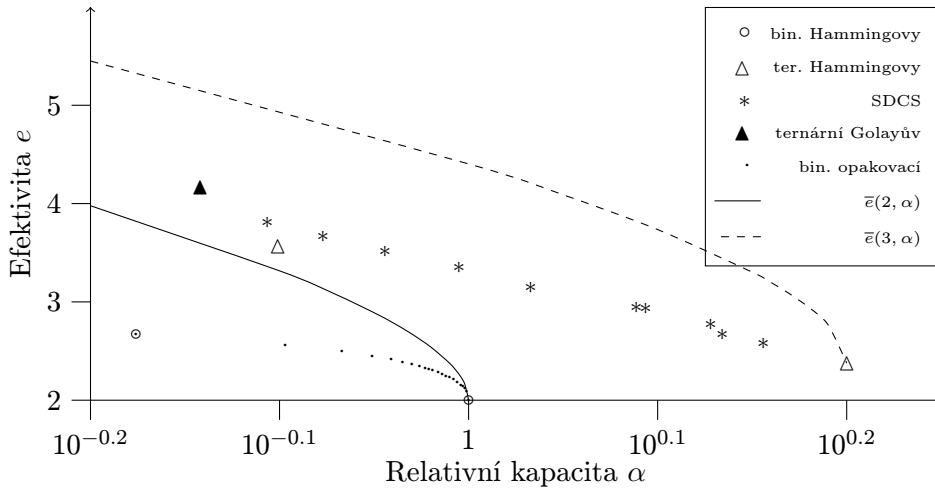
2 najdi s_1, \dots, s_n takové, že $g = \sum_{i=1}^n s_i a_i$ a zároveň $\sum_{i=1}^n |s_i| = w_{\mathcal{A}}(g)$

3 **return** $(x_1 + s_1, \dots, x_n + s_n)$

Důkaz. $\text{Ext}_{\mathcal{A}}(y_1, \dots, y_n) = \sum_{i=1}^n (x_i + s_i) a_i = g + \sum_{i=1}^n x_i a_i = z.$ □

(n, r, G)	α	e	\mathcal{A}
$(4, 3, \mathbb{Z}_{53})$	1,4320	2,5727	{1, 2, 6, 18}
$(3, 2, \mathbb{Z}_{17})$	1,3625	2,6726	{1, 2, 6}
$(5, 3, \mathbb{Z}_{105})$	1,3428	2,7756	{1, 3, 14, 36, 42}
$(6, 3, \mathbb{Z}_{174})$	1,2405	2,9300	{1, 3, 9, 21, 51, 86}
$(4, 2, \mathbb{Z}_{30})$	1,2267	2,9441	{1, 3, 9, 14}
$(5, 2, \mathbb{Z}_{42})$	1,0785	3,1445	{1, 2, 7, 14, 18}
$(6, 2, \mathbb{Z}_{61})$	0,9885	3,3498	{1, 2, 5, 11, 19, 27}
$(7, 2, \mathbb{Z}_{80})$	0,9031	3,5122	{1, 22, 26, 30, 34, 36, 39}
$(8, 2, \mathbb{Z}_{104})$	0,8376	3,6676	{2, 4, 6, 13, 16, 34, 39, 40}
$(9, 2, \mathbb{Z}_{132})$	0,7827	3,8109	{2, 11, 33, 34, 44, 50, 55, 58, 62}

TABULKA 3.2: Příklady různých SDCS, jejich relativní kapacita a efektivita. Převzato z [9].



OBRÁZEK 3.1: Efektivita SDCS z tabulky 3.2 v závislosti na relativní kapacitě ve srovnání s perfektními kódy a s horní mezí na spodní efektivitu pro $q = 2$ a 3 .

459 V případě, že prvky stegoobjektu mají omezený obor hodnot $\mathcal{X} \subseteq \mathbb{Z}$, může i zde nastat
460 stejný problém jako při maticovém vkládání, že $y_i = x_i - 1 \notin \mathcal{X}$ nebo $y_i = x_i + 1 \notin \mathcal{X}$.
461 V prvním případě musíme situaci napravit tím, že zvolíme $y_i = x_i + 2$ a ve druhém případě
462 $y_i = x_i - 2$. Tyto speciální případy z následujících úvah vynecháváme.

463 Z algoritmu vkládání vidíme, že hodnota r udává horní mez na počet změn v bloku
464 vyvolaných SDCS vkládáním. Relativní kapacita a efektivita SDCS vkládání jsou zřejmě

$$\alpha = \frac{\log_2 |G|}{n} \quad \text{a} \quad e = \frac{\log_2 |G|}{\sum_{g \in G} w_{\mathcal{A}}(g)/|G|} = \frac{|G|\alpha n}{\sum_{g \in G} w_{\mathcal{A}}(g)},$$

465 za předpokladu, že zprávy jsou voleny z G náhodně s rovnoměrným rozdělením. Ta-
466 bulka 3.2 uvádí příklady několika různých SDCS, jejich relativní kapacitu a efektivitu.
467 Na obrázku 3.1 potom vidíme srovnání těchto SDCS s perfektními kódy a s horní mezí na
468 spodní efektivitu pro $q = 2$ a 3 .

469 Sčítáním odečítáním i prvků z \mathcal{A} lze sestavit nejvýše $2^i \binom{n}{i}$ prvků z G , proto pro každou
470 (n, r, G) -SDCS platí

$$|G| \leq \sum_{i=0}^r 2^i \binom{n}{i} = V_3(n, r). \quad (3.1)$$

471 Z tohoto můžeme odvodit obdobu věty 2.7.

472 **Věta 3.2.** *Pro každou (n, r, G) -SDCS platí $r \geq nH_3^{-1}(\alpha/\log_2 3)$.*

473 *Důkaz.* Jestliže $r/n \geq \frac{2}{3}$ pak tvrzení věty platí triviálně. Dále tedy předpokládejme, že
474 $r/n < \frac{2}{3}$. Podle rovnice 3.1 a lemmatu 2.6 je $\log_3 |G| \leq \log_3 V_3(n, r) \leq nH_3(r/n)$. Vzhledem
475 k tomu, že H_3 je na intervalu $[0, \frac{2}{3}]$ rostoucí, máme $H_3^{-1}((\log_3 |G|)/n) \leq r/n$. Zbývá dosadit
476 $\log_3 |G| = n\alpha/\log_2 3$. \square

477 Stejně jako v části o maticovém vkládání můžeme definovat pojem spodní efektivity
478 tak, že očekávaný počet změn vyvolaných vkládáním nahradíme maximálním počtem
479 změn. Z předchozí věty pak plyne horní odhad na spodní efektivitu vkládání

$$e := \frac{\alpha n}{r} \leq \frac{\alpha}{H_3^{-1}(\alpha/\log_2 3)}.$$

480 Tento výsledek je shodný s tím, který jsme získali pro maticové vkládání ternárními kódy.
481 Jinými slovy pomocí SDCS nelze v nejhorším případě dosáhnout o nic lepší efektivity
482 než pomocí maticového vkládání. Nicméně SDCS mohou vést k mnohem jednodušší sché-
483 matům. Dobrým příkladem je $(3, 2, \mathbb{Z}_{17})$ -SDCS $\{1, 2, 6\}$ s relativní kapacitou 1,3625 a
484 efektivitou 2,6726. Zkuste sestrojít srovnatelně jednoduchý ternární kód s podobnými pa-
485 rametry.

486

4. PSANÍ NA MOKRÝ PAPIR

487 V kapitole 2 jsme viděli, jak se dá minimalizovat dopad změn vyvolaných vkládáním, a
488 to minimalizací jejich počtu pomocí samoopravných kódů. V této kapitole si předvedeme
489 další aplikaci samoopravných kódů, kterou lze opět využít k minimalizaci dopadu změn,
490 ale také k několika dalším zajímavým účelům.

491 Je zřejmé, že vkládání do některých prvků nosiče má vyšší dopad na detekovatelnost
492 než vkládání do jiných. V případě rastrových a paletových obrázků se například chceme
493 vyhnout vkládání do pixelů, které se nacházejí v oblastech s víceméně uniformní barvou,
494 zejména pak v přesvícených oblastech obrázku. V obrázcích JPEG se zase chceme vyhnout
495 vkládání do nulových AC koeficientů, protože jejich změna zanechává v obrázku viditelné
496 stopy.

497 Máme-li nosič délky n a zprávu délky $m \leq n$, pak můžeme podle předchozích kritérií
498 vybrat m prvků nosiče, které jsou nejvhodnější pro vkládání, a do nich vložit zadanou
499 zprávu. Problém je, že příjemce nemůže vědět, do kterých prvků jsme vkládali. U obrázku
500 JPEG jsme se mohli například vyhýbat nulovým AC koeficientům, ale při vkládání se
501 některé nenulové koeficienty změnily na nulové a příjemce neví, které nuly vznikly vklá-
502 dáním, a které pocházejí z nosiče. Obdobně u rastrového nebo paletového obrázku jsme se
503 mohli vyhýbat oblastem s nízkou texturou, ale vložením se textura v některých oblastech
504 snížila a příjemce opět nemůže vědět, ve kterých oblastech se textura snížila vkládáním, a
505 ve kterých byla nízká již v nosiči. Tento problém se dá řešit tak, že dojde-li například při
506 vkládání do obrázku JPEG k vynulování AC koeficientu, pak vložený bit považujeme za
507 ztracený, vynulovaný koeficient ponecháme na hodnotě 0 a bit vložíme znovu do dalšího
508 nenulového AC koeficientu. Příjemce potom extrahuje bity zprávy výhradně z nenulových
509 AC koeficientů. Nevýhodou tohoto postupu je, že v typickém obrázku JPEG tímto dojde
510 ke snížení kapacity o přibližně 25 %. Tento problém by se dal také řešit tak, že v rámci
511 vkládání bychom příjemci navíc nějakým způsobem poskytli informaci o tom, které prvky
512 obsahují vloženou zprávu a které nikoliv. Měli bychom zdůraznit, že tuto informaci není

513 obecně možné sdělit v rámci klíče, protože ten si strany zpravidla dohodnou ještě předtím,
514 než vůbec dojde k výběru nosiče nebo dokonce vzniku nosiče.

515 Tento problém se přirovnává k psaní na mokrý papír. Na papíře máme obrázek, jehož
516 některé pixely jsou mokré a ostatní suché. Suché pixely smíme upravovat, ale mokré nikoliv.
517 Obrázek odešleme, ale ještě než dorazí příjemci, tak uschne. Příjemce nyní netuší, které
518 pixely jsme využili ke vkládání, a které nikoliv.

519 Podle následující věty můžeme i prostřednictvím částečně mokrého nosiče skoro jistě
520 sdělit příjemci téměř stejně velkou zprávu, jako kdyby příjemce znal rozmístění suchých
521 prvků. Podmínkou je dostatečná velikost nosiče. Poznamenejme, že příjemce se ve skuteč-
522 nosti nedozví, které prvky byly suché.

523 **Věta 4.1** (o mokrém nosiči). *Pro každé $\varepsilon > 0$, $\delta > 0$ a $\sigma \in [0, 1]$ existuje $n \in \mathbb{N}$ a stegosys-
524 tém takový, že do nosiče velikosti n symbolů, z nichž σn je suchých, lze s pravděpodobností
525 alespoň $1 - \delta$ vložit informaci velikosti $(\sigma - \varepsilon)n$ symbolů.*

526 *Důkaz.* Označme množinu všech symbolů Σ a její velikost q . (Obvykle uvažujeme $\Sigma = \mathbb{F}_2$
527 nebo $\Sigma = \mathbb{F}_3$.) Dále označme \mathcal{Z} množinu všech zpráv, ta má $q^{(\sigma - \varepsilon)n}$ prvků. Stegosystém
528 se inicializuje vytvořením kódové knihy $\mathcal{B} = \{\mathcal{P}_z \mid z \in \mathcal{Z}\}$, do které se náhodně rozmístí
529 všechny n -tice ze Σ^n tak, aby na každé stránce \mathcal{P}_z bylo $q^{(1 - \sigma + \varepsilon)n}$ n -tic. Vkládací a extrakční
530 algoritmus tuto kódovou knihu sdílí. Vkládací algoritmus má na vstupu nosič s $n - \sigma n$
531 mokřými symboly a zprávu z . Vkládání probíhá tak, že algoritmus se pokusí na stránce \mathcal{P}_z
532 najít vhodnou n -tici, tj. takovou, která se na mokřích pozicích shoduje s nosičem, a
533 v případě úspěchu vytvoří stegoobjekt zaznamenáním této n -tice do nosiče. Mokré pozice
534 tak zůstávají nezměněny. Extrakční algoritmus přečte ze stegoobjektu n -tici a vyhledá ji
535 v knize, čímž určí stránku, na které se nachází, a z indexu stránky získá zprávu. Otázkou
536 je, zda se vkládacímu algoritmu podaří na stránce \mathcal{P}_z najít n -tici, která má na $n - \sigma n$
537 předem určených pozicích předem určené hodnoty. Pro každou zprávu je v množině Σ^n
538 celkem $q^{\sigma n}$ vhodných n -tic. Stránka \mathcal{P}_z vznikla náhodným výběrem $q^{(1 - \sigma + \varepsilon)n}$ n -tic ze
539 Σ^n . Pravděpodobnost, že náhodně zvolená n -tice z Σ^n není vhodná je $(q^n - q^{\sigma n})/q^n$.
540 Pravděpodobnost, že žádná z $q^{n - \sigma n + \varepsilon n}$ náhodně vybraných n -tic, které leží v \mathcal{P}_z , není
541 vhodná, je méně než

$$\left(\frac{q^n - q^{\sigma n}}{q^n}\right)^{q^{n - \sigma n + \varepsilon n}} = \left(\left(1 + \frac{-1}{q^{n(1 - \sigma)}}\right)^{q^{n(1 - \sigma)}}\right)^{q^{\varepsilon n}}.$$

542 Pro $n \rightarrow \infty$ má výraz $\left(1 + \frac{-1}{q^{n(1 - \sigma)}}\right)^{q^{n(1 - \sigma)}}$ limitu e^{-1} a pro dostatečně velká n je tedy
543 menší než 1. Z toho plyne, že pravděpodobnost, že \mathcal{P}_z neobsahuje vhodnou n -tici, se
544 s rostoucím n blíží nule. \square

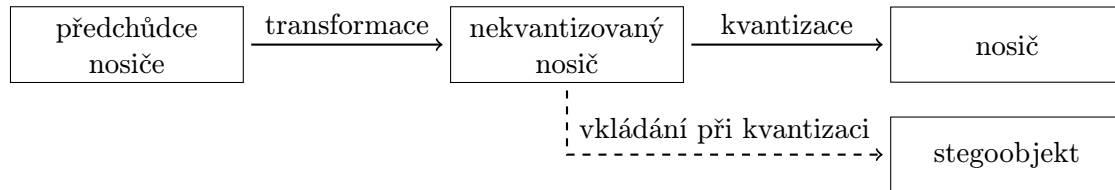
545 Pro algoritmy vkládání a extrakce popsané v důkazu předchozí věty můžeme použít
546 označení $\mathbf{y} = \text{Emb}_{\mathcal{B}}(\mathbf{x}, \mathcal{S}, \mathbf{z})$ a $\mathbf{z} = \text{Ext}_{\mathcal{B}}(\mathbf{y})$, kde \mathbf{x} , \mathbf{y} , \mathbf{z} a \mathcal{S} jsou nosič, stegoobjekt,
547 zpráva a množina indexů suchých složek nosiče. Je zřejmé, že stegosystém tak, jak je
548 popsán v důkazu, není vhodný pro praktické použití, protože velikost kódové knihy je
549 exponenciální v n . Tento problém lze řešit pomocí samoopravných kódů, a to tak, že
550 kódovou knihou bude faktorprostor $\mathcal{B} = \mathbb{F}_q^n / \mathcal{C}$ a stránky knihy budou jeho třídy $\mathcal{P}_z =$
551 $\mathcal{C}(\mathbf{z})$. Potom $\text{Ext}_{\mathbb{F}_q^n / \mathcal{C}}(\mathbf{y}) = \mathbf{H}\mathbf{y}$, kde \mathbf{H} je paritní matice kódu \mathcal{C} . Podrobněji toto řešení
552 rozebereme na konci kapitoly, nejdříve si ale ukážeme několik aplikací.

553 4.1. ZÁKLADNÍ APLIKACE PSANÍ NA MOKRÝ PAPIR

554 V úvodu této kapitoly jsme byli motivováni zabývat se psáním na mokrý papír, protože
555 jsme usilovali o snížení detekovatelnosti vkládání tím, že ke vkládání využijeme ty prvky

556 nosiče, jejichž změna má nejnižší dopad na detekovatelnost. Nyní se podíváme na dvě
557 metody publikované v článku [6], které na této myšlence budují.

558 **4.1.1. Vkládání při kvantizaci.** Uvažme situaci, kdy nosič vzniká z nějakého předchůdce
559 transformací a následnou kvantizací. Transformací zde rozumíme nějaký proces jehož vý-
560 sledkem jsou obvykle hodnoty s plovoucí čárkou. Kvantizací se rozumí zaokrouhlení každé
561 z těchto hodnot na nejbližší povolenou hodnotu. Typickou množinou povolených hodnot
562 může být $\{0, 1, \dots, 255\}$, potom kvantizace převádí například $-3 \mapsto 0$ nebo $4,71 \mapsto 5$.
563 Množina povolených hodnot nemusí být pouze interval celočíselných hodnot, ale může se
564 například jednat o množinu všech celočíselných násobků čísla 3.



565

566 Pod pojmem transformace si lze představit zejména následující.

- 567 • Předchůdce je rastrový obrázek a transformace snižuje hloubku barev obrázku tak,
568 že každý prvek vydělí nastavenou konstantou. Redukujeme-li například 30bitový
569 obrázek (tj. na každou červenou, zelenou nebo modrou složku připadá 10 bitů), na
570 24bitový obrázek (tj. na složku připadá 8 bitů), dělíme každý prvek číslem 4.
- 571 • Předchůdce je rastrový obrázek a transformace zmenšuje rozměry obrázku. Například
572 zmenšení na polovinu v obou směrech lze provést tak, že každý blok 2×2 pixelů se
573 převede na jediný pixel zprůměrováním barev všech čtyř pixelů. (Mnohem lepších
574 vizuálních výsledků lze však dosáhnout nejrůznějšími interpolačními metodami.)
- 575 • Předchůdce je rastrový obrázek a transformace převádí každý blok 8×8 hodnot
576 jasové složky obrázku na blok 8×8 DCT koeficientů.

577 U obrázků, které vznikly takovýmto procesem, získáváme měřítko, podle kterého lze
578 vybírat prvky nosiče vhodné ke vkládání. Nechť \mathbb{Z} je množina povolených hodnot. Má-li
579 nezaokrouhlený prvek například hodnotu 14,9, pak při zaokrouhlení na nejbližší celé číslo
580 vzniká zaokrouhlovací chyba 0,1 a distorze $0,1^2$. Kdybychom do takového prvku chtěli
581 vložit bit 0, zaokrouhlili bychom jej na 14, a naopak při vkládání bitu 1 bychom zaokrouhlili
582 na 15. Očekávaná distorze při vkládání do takového prvku je tedy $(0,1^2 + 0,9^2)/2 = 0,41$,
583 což je o 0,4 vyšší než při pouhém zaokrouhlení. Srovnáme to se situací, kdy nezaokrouhlený
584 prvek má hodnotu 14,5. Je zřejmé, že očekávaná distorze vyvolaná vkládáním do takového
585 prvku je shodná s distorzí vyvolanou zaokrouhlením na nejbližší celé číslo, a to 0,25.
586 Takovéto prvky jsou tedy zdaleka nejvhodnější pro vkládání. Čím vyšší je vzdálenost
587 nezaokrouhleného prvku od množiny celých čísel, tím vhodnější je daný prvek pro vkládání.

588 Obecně můžeme ke každému prvku nezaokrouhleného nosiče \tilde{x}_i přiřadit číslo ρ_i , které
589 udává rozdíl mezi očekávanou distorzí vyvolanou vkládáním do prvku a distorzí vyvolanou
590 zaokrouhlením na nejbližší povolenou hodnotu. Množinu indexů suchých prvků nosiče \mathcal{S}
591 zvolíme tak, aby celkový očekávaný příspěvek vkládání k distorzí $\sum_{i \in \mathcal{S}} \rho_i$ byl minimalizo-
592 ván. V případě, kdy množina povolených hodnot je \mathbb{Z} , můžeme ρ_i vyjádřit pomocí vzdále-
593 nosti \tilde{x}_i od množiny celých čísel $\delta_i = \text{dist}(\tilde{x}_i, \mathbb{Z}) = |\lceil \tilde{x}_i \rceil - \tilde{x}_i|$ jako $\rho_i = \frac{1}{2}(\delta_i^2 + (1 - \delta_i)^2) - \delta_i^2 =$
594 $\frac{1}{2} - \delta_i$.

595 Tento postup, kdy nejdříve shora uvedeným způsobem vybereme suché prvky nosiče
596 a potom metodou psaní na mokřý papír vložíme zprávu tak, že usměrníme jejich zao-
597 krouhlení, nazýváme *vkládání při kvantizaci* (perturbed quantization, PQ).

598 **4.1.2. Vkládání při dvojitě ztrátové kompresi.** V předchozí části jsme poznamenali,
 599 že zdaleka nejvýhodnější je vkládat do prvků, jejichž nezaokrouhlená hodnota leží přesně
 600 uprostřed mezi dvěma po sobě následujícími povolenými hodnotami. Takoveto prvky bu-
 601 deme označovat jako *nestranné*. V běžném obrázku bude takovýchto prvků jen velmi málo.
 602 Otázkou je, zda by se dalo nějakým přirozeným způsobem zařídit, aby se takovýchto prvků
 603 vyskytovalo víc.

604 Pripomeňme, jakým způsobem se ve formátu JPEG docílí ztrátové komprese obrázku.
 605 Blok 8×8 hodnot (např. hodnot jasové složky obrázku) se převede na blok 8×8 DCT
 606 koeficientů $\mathbf{d} = (d_{i,j})$ a tyto koeficienty se následně zaokrouhlí s různou mírou přesnosti.
 607 Přesnost zaokrouhlení každého DCT koeficientu je určena kvantizační tabulkou $\mathbf{Q} = (q_{i,j})$.
 608 Pro každý DCT koeficient se spočítá podíl $d_{i,j}/q_{i,j}$, a ten se zaokrouhlí na nejbližší celé
 609 číslo. Při otevírání obrázku se naopak každé z těchto zaokrouhlených čísel vynásobí přísluš-
 610 nou hodnotou $q_{i,j}$, čímž opět získáme DCT koeficient. Tuto fázi komprese můžeme tedy
 611 charakterizovat jako zaokrouhlení každého DCT koeficientu $d_{i,j}$ na nejbližší $q_{i,j}$ -násobek.

612 Jeden způsob, jak zajistit existenci nestranných prvků, je dvakrát opakovaná ztrátová
 613 komprese obrázku JPEG. Začneme tím, že vytvoříme anebo vezmeme existující obrázek
 614 JPEG (předchůdce nosiče). Transformace tohoto předchůdce spočívá v tom, že obrázek
 615 dekomprimujeme a znovu komprimujeme, ale tentokrát v nižší kvalitě. Při této kompri-
 616 maci se můžeme za určitých okolností setkat s poměrně velkým počtem nestranných DCT
 617 koeficientů. Velice dobrých výsledků lze například dosáhnout použitím standardních kvan-
 618 tizačních tabulek s faktory kvality 85 a 70

$$Q^{(85)} = \begin{pmatrix} 5 & 3 & \boxed{3} & 5 & 7 & 12 & 15 & 18 \\ 4 & 4 & 4 & 6 & 8 & 17 & 18 & 17 \\ 4 & 4 & 5 & 7 & 12 & 17 & 21 & 17 \\ 4 & 5 & 7 & 9 & 15 & 26 & 24 & 19 \\ 5 & 7 & 11 & 17 & 20 & 33 & 31 & 23 \\ 7 & 11 & 17 & 19 & 24 & 31 & 34 & 28 \\ 15 & 19 & 23 & 26 & 31 & 36 & 36 & 30 \\ 22 & 28 & 29 & 29 & 34 & 30 & 31 & 30 \end{pmatrix}, \quad Q^{(70)} = \begin{pmatrix} 10 & 7 & \boxed{6} & 10 & 14 & 24 & 31 & 37 \\ 7 & 7 & 8 & 11 & 16 & 35 & 36 & 33 \\ 8 & 8 & 10 & 14 & 24 & 34 & 41 & 34 \\ 8 & 10 & 13 & 17 & 31 & 52 & 48 & 37 \\ 11 & 13 & 22 & 34 & 41 & 65 & 62 & 46 \\ 14 & 21 & 33 & 38 & 49 & 62 & 68 & 55 \\ 29 & 38 & 47 & 52 & 62 & 73 & 72 & 61 \\ 43 & 55 & 57 & 59 & 67 & 60 & 62 & 59 \end{pmatrix}.$$

619 Předchůdce je komprimován s faktorem kvality 85, zatímco nosič a stegoobjekt s fakto-
 620 rem kvality 70. Podívejme se například na zarámované kvantizační koeficienty v obou
 621 tabulkách. Předchůdce nosiče je komprimován tak, že DCT koeficienty na této pozici jsou
 622 zaokrouhlovány na nejbližší celočíselný násobek čísla 3. Při snížení kvality obrázku se DCT
 623 koeficienty na této pozici zaokrouhlují na nejbližší celočíselný násobek čísla 6. To znamená,
 624 že téměř polovina bloků bude mít na této pozici nestranný DCT koeficient, který může být
 625 zaokrouhlen jak nahoru, tak dolů, aniž by vyvolával větší podezření. Na pozicích, kde se
 626 používají větší kvantizační koeficienty, bude nestranných DCT koeficientů výrazně méně
 627 než polovina, protože hodně z nich bude v předchůdci nosiče zaokrouhleno na 0.

628 Dvojice kvantizačních tabulek $Q^{(85)}$ a $Q^{(70)}$ dosahuje dobrých výsledků proto, že se v ní
 629 vyskytuje mnoho pozic, které mají tendenci dávat vzniknout nestranným koeficientům.
 630 Tyto pozice jsou vyznačeny tučně a můžeme si všimnout, že se mezi tabulkami vždy
 631 liší o násobek 2. To ovšem není nutnou podmínkou pro to, aby pozice dávala vzniknout
 632 nestranným koeficientům. Máme-li dvojici kvantizačních matic $Q^{(f_1)}$ a $Q^{(f_2)}$, pak na pozici
 633 (i, j) mohou při dvojitě ztrátové kompresi vznikat nestranné DCT koeficienty právě tehdy,
 634 když existují $a, b \in \mathbb{Z}$ takové, že $aq_{i,j}^{(f_1)} = bq_{i,j}^{(f_2)} + \frac{1}{2}q_{i,j}^{(f_2)}$.

635 Na první pohled se může zdát, že dvojitě komprimované obrázky JPEG budou vyvolá-
 636 vat podezření, ale takové obrázky mohou ve skutečnosti snadno vznikat přirozeně. Běžný
 637 uživatel má fotoaparát, jehož výstupem jsou obrázky JPEG. Obrázek pak může otevřít

638 v editoru a provést na něm některé běžné úpravy jako například odstranění efektu červe-
639 ných očí. Po dobu editace je zpravidla nutné převést obrázek do rastrového formátu a poté
640 znovu uložit do formátu JPEG. Výjimku tvoří úpravy jako rotace obrázku o násobek 90° a
641 souměrnost podle osy x nebo y , které lze provést přímo ve formátu JPEG. Při opětovném
642 ukládání se může stát, že obrázek je uložen v nižší kvalitě než originál, a to buď záměrně,
643 aby se ušetřilo místo, anebo protože grafický editor implicitně ukládá obrázky v kvalitě,
644 která je nižší než kvalita nastavená na fotoaparátu.

645 Při vkládání do takto vytvořených nestranných DCT koeficientů je poměrně obtížné
646 rozeznat stegoobjekty od nosičů, které vznikly prostou dvojitou kompresí. Odhalení ste-
647 goobjektů může být usnadněno ve chvíli, kdy se pro vkládání použijí bloky s více méně
648 uniformní barvou. Nejlepších výsledků tedy dosáhneme, když do množiny suchých prvků
649 zahrneme pouze ty nestranné koeficienty, které leží v nejvíce neuniformních blocích před-
650 chůdce nosiče. Uniformita bloku se zpravidla měří dvěma způsoby:

651 **Textura bloku** Blok DCT koeficientů dekomprimujeme na blok 8×8 hodnot. Tento blok
652 rozdělíme na 16 podbloků velikosti 2×2 . V každém podbloku změříme rozdíl mezi
653 nejmenší a největší hodnotou. Textura bloku je součet všech 16 rozdílů.

654 **Energie bloku** Energie bloku je součet čtverců všech kvantizovaných DCT koeficientů
655 v bloku.

656 Shrňme shora uvedený postup. Máme zprávu délky m a potřebujeme zvolit nějakých $s > m$
657 suchých prvků v nekvantizovaném nosiči. Ty zvolíme tak, že vyhledáme nejméně uniformní
658 blok v předchůdci nosiče, ten rekomprimujeme na nižší kvalitu a přitom získáme blok ne-
659 kvantizovaného nosiče s určitým počtem nestranných DCT koeficientů, které zahrneme
660 do množiny suchých prvků. Tento proces opakujeme na dalších nejméně uniformních blo-
661 cích v předchůdci nosiče tak dlouho, dokud množina suchých prvků nedosáhne požadované
662 velikosti. Zbylé bloky také rekomprimujeme na nižší kvalitu, ale všechny jejich prvky ozna-
663 čujeme jako mokré. Zprávu potom vložíme metodou vkládání při kvantizaci. Tento postup
664 se nazývá *texture-adaptive perturbed quantization* (PQt) nebo *energy-adaptive perturbed*
665 *quantization* (PQe) podle toho, které měřítko uniformity bloku použijeme.

666 **4.1.3. Modifikované maticové kódování.** Modifikované maticové kódování (modified
667 matrix encoding, MME) [8] ve skutečnosti nevyužívá psaní na mokřý papír, ale jedná se
668 o variaci na vkládání při kvantizaci, a proto jej uvádíme zde.

669 Stegosystém MME je podobný stegosystému F5 v tom, že vkládá do nenulových AC ko-
670 eficientů obrázku JPEG a je založen na použití binárního Hammingova kódu. Označme \mathbf{H}
671 paritní matici Hammingova $[n, n-m]_2$ kódu délky $n = 2^m - 1$. K extrakci se používá tentýž
672 algoritmus jako u stegosystému F5 pouze s tím rozdílem, že parita negativních koeficientů
673 se neobrací. Jinými slovy jestliže $\mathbf{y} \in \mathbb{F}_2^n$ je blok nejnižších bitů nenulových kvantizovaných
674 AC koeficientů stegoobrázku, pak blok zprávy získáme jako $\mathbf{z} = \text{Ext}_{\mathbf{H}}(\mathbf{y}) = \mathbf{H}\mathbf{y} \in \mathbb{F}_2^m$.

675 MME vkládání se od F5 vkládání liší ve čtyřech aspektech.

- 676 • Zatímco při F5 vkládání se změny v nosiči provádějí snížením absolutní hodnoty ko-
677 eficientu, při MME vkládání se využívá znalost nekvantizovaného nosiče a případná
678 změna se provede zaokrouhlením na druhé nejbližší nenulové celé číslo.
- 679 • Při MME vkládání nedochází ke smrštění, tj. $1 \mapsto 0$ nebo $-1 \mapsto 0$. Je-li potřeba
680 změnit koeficient s hodnotou 1 nebo -1 na sudou hodnotu, použije se vždy $1 \mapsto 2$ a
681 $-1 \mapsto -2$.

- Zatímco stegosystém F5 asociuje záporné liché AC koeficienty s hodnotou 0 a záporné sudé koeficienty s hodnotou 1, aby se vyvážílo smršťování, stegosystém MME jednoduše asociuje všechny AC koeficienty se zbytkem po dělení 2.
- Zatímco F5 vkládání provádí nejvýše jednu změnu v každém bloku nosiče, MME vkládání provádí nejvýše d změn v každém bloku nosiče tak, aby se minimalizovala celková odchylka stegoobjektu od nezaokrouhleného nosiče. Hovoříme o MME d vkládání.

Poslední bod rozeberme podrobněji. Označme $\mathbf{x} \in \mathbb{F}_2^n$ blok nejnižších bitů nenulových kvantizovaných AC koeficientů nosiče. Podobně jako v části 4.1.1 přiřadíme každé pozici i číslo ρ_i , které bude tentokrát udávat, o kolik se navýší odchylka od nezaokrouhleného nosiče při změně na pozici i . Označme $\tilde{x}_i \in \mathbb{R}$ hodnotu příslušného nezaokrouhleného prvku a zaokrouhlovací chybu $\delta_i := |[\tilde{x}_i] - \tilde{x}_i|$. Pokud $|\tilde{x}_i| > 1$, pak chyba při zaokrouhlení na druhé nejbližší nenulové celé číslo je $1 - \delta_i$. To znamená, že při změně bitu x_i se odchylka od nezaokrouhleného nosiče navýší o $\rho_i = 1 - 2\delta_i$. Pokud $\tilde{x}_i \in [-1, -\frac{1}{2}] \cup [\frac{1}{2}, 1]$, pak druhé nejbližší celé číslo je -2 nebo 2 a zaokrouhlením na ně vzniká chyba $1 + \delta_i$. To znamená, že při změně bitu x_i se odchylka od nezaokrouhleného nosiče navýší o $\rho_i = 1$. Připomeňme, že $\tilde{x}_i \notin (-\frac{1}{2}, \frac{1}{2})$, jinak by totiž $[\tilde{x}_i] = 0$, ale takové AC koeficienty při vkládání i extrakci přeskakujeme.

Jakmile máme spočítány hodnoty ρ_i , zjistíme všechny způsoby, jak nejvýše d změnami vložit zprávu \mathbf{z}

$$\mathcal{E} = \{ \mathbf{e} \in \mathbb{F}_2^n \mid \mathbf{H}\mathbf{e} = \mathbf{z} - \mathbf{H}\mathbf{x}, w(\mathbf{e}) \leq d \}.$$

Použijeme ten vektor změn, který minimalizuje odchylku stegoobjektu od nezaokrouhleného nosiče

$$\mathbf{y} = \mathbf{x} + \arg \min_{\mathbf{e} \in \mathcal{E}} \sum_{\substack{1 \leq i \leq n \\ e_i = 1}} \rho_i.$$

V praxi má smysl použít algoritmus MME2 nebo MME3. Pro $d > 3$ se totiž dosahuje jen zanedbatelně lepších výsledků. Všimněme si, že konstrukce množiny \mathcal{E} se díky povaze binárních Hammingových kódů obejde bez jakékoliv práce s maticemi. Proto má MME2 časovou složitost $O(n)$ a MME3 $O(n^2)$.

4.2. DALŠÍ APLIKACE PSANÍ NA MOKRÝ PAPÍR

4.2.1. Dvouúrovňové ± 1 vkládání. S ± 1 vkládáním jsme se již jednou setkali jakožto s efektivní obranou proti kvantitativním útokům na vkládání do nejnižšího bitu. Připomeňme, že ± 1 vkládání zprávy $\mathbf{z} \in \{0, 1\}^m$ do nosiče $\mathbf{x} \in \mathbb{Z}^n$ funguje podle následujícího pravidla. Jestliže $\text{LSB}(x_i) = z_i$, pak přiřadíme $y_i := x_i$, v opačném případě zvolíme náhodně $a \in \{-1, 1\}$ a přiřadíme $y_i := x_i + a$. Z následující tabulky vidíme, že rozhodnutí přičíst anebo odečíst 1 nám dává plnou kontrolu nad hodnotou druhého nejnižšího bitu. Radši než volit změny a náhodně, můžeme je využít k vložení další zprávy do nosiče pomocí psaní na mokrý papír. Této technice říkáme *dvouúrovňové ± 1 vkládání* [11].

Poslední dvě cifry binárního rozvoje

	x_i	$(\dots 00)_2$	$(\dots 01)_2$	$(\dots 10)_2$	$(\dots 11)_2$
$x_i + 1$		$(\dots 01)_2$	$(\dots 10)_2$	$(\dots 11)_2$	$(\dots 00)_2$
$x_i - 1$		$(\dots 11)_2$	$(\dots 00)_2$	$(\dots 01)_2$	$(\dots 10)_2$

Nechť \mathbf{H} je paritní matice $[n, n - m_1]_2$ kódu \mathcal{C} s průměrnou vzdáleností r_a a \mathcal{B} je kódová kniha, která umožňuje vkládat m_2 bitů do nosiče délky n bitů, z nichž r_a je suchých. Označme $\mathbf{x}' = (\text{LSB}(x_1), \dots, \text{LSB}(x_n))^T$ vektor nejnižších bitů nosiče \mathbf{x} a \mathbf{x}''

721 vektor druhých nejnižších bitů nosiče. Obdobně značíme také nejnižší a druhé nejnižší
722 bity \mathbf{y}' a \mathbf{y}'' stegoobjektu \mathbf{y} . Zpráva je rozdělena na dvě části $\mathbf{z}' \in \mathbb{F}_2^{m_1}$ a $\mathbf{z}'' \in \mathbb{F}_2^{m_2}$.
723 Dvouúrovňovou extrakci lze vyjádřit jednoduše jako $\mathbf{z}' = \text{Ext}_{\mathbf{H}}(\mathbf{y}')$ a $\mathbf{z}'' = \text{Ext}_{\mathcal{B}}(\mathbf{y}'')$.

724 Dvouúrovňové ± 1 vkládání má dvě fáze. V první fázi zjistíme jaké změny bude třeba
725 provést na nejnižších bitech nosiče, čili spočteme $\mathbf{y}' = \text{Emb}_{\mathbf{H}}(\mathbf{x}', \mathbf{z}')$. Na pozicích, kde se
726 budou provádět změny jsme navíc schopni ovlivnit druhý nejnižší bit. Tyto pozice tedy
727 označíme jako suché $\mathcal{S} = \{i \mid x'_i \neq y'_i\}$. V druhé fázi použijeme metodu psaní na mokrý
728 papír a spočteme $\mathbf{y}'' = \text{Emb}_{\mathcal{B}}(\mathbf{x}'', \mathcal{S}, \mathbf{z}'')$. Stegoobjekt \mathbf{y} se vytvoří z nosiče \mathbf{x} změnami $+1$
729 nebo -1 tak, aby \mathbf{y}' byly nejnižší bity a \mathbf{y}'' druhé nejnižší bity stegoobjektu. Očekávaný
730 počet těchto změn je r_a a tedy také očekávaná velikost množiny \mathcal{S} je r_a .

Označíme-li e efektivitu kódu \mathcal{C} , pak $r_a = m_1/e$. Podle věty o mokrém nosiči je $m_2 \approx |\mathcal{S}| \approx r_a$. Relativní kapacita a efektivita dvouúrovňového ± 1 vkládání je tedy

$$\alpha_{\pm 1} = \frac{m_1 + m_2}{n} \approx \frac{m_1 + m_1/e}{n} = \alpha + \frac{\alpha}{e}, \quad (4.1)$$

$$e_{\pm 1} = \frac{m_1 + m_2}{r_a} \approx \frac{m_1 + m_1/e}{m_1/e} = e + 1. \quad (4.2)$$

731 Zároveň si můžeme všimnout, že

$$\frac{\alpha}{e} = \frac{r_a}{n} = \frac{\alpha_{\pm 1}}{e_{\pm 1}}. \quad (4.3)$$

732 Vkládání se změnami ± 1 si spojujeme s vkládáním pomocí ternárních kódů. Jak uka-
733 zuje následující tvrzení, z hlediska efektivity dosahuje dvouúrovňové vkládání stejně dob-
734 rých výsledků jako ternární vkládání. Oproti ternárním kódům má dvouúrovňové ± 1 vklá-
735 dání jednu výhodu. Při použití ternárních kódů je totiž nutné převádět mezi ternární
736 reprezentací zpráv a obvyklou binární reprezentací, což zde odpadá.

737 **Tvrzení 4.2.** *Nechť \mathcal{C} je binární kód, jehož efektivita dosahuje horní meze na spodní*
738 *efektivitu binárních kódů. Potom dvouúrovňovým ± 1 vkládáním s kódem \mathcal{C} lze dosáhnout*
739 *efektivity libovolně blízké horní mezi na spodní efektivitu ternárních kódů, za předpokladu,*
740 *že kód \mathcal{C} je dostatečně dlouhý.*

741 *Důkaz.* Dle předpokladu je $e = \alpha/H_2^{-1}(\alpha)$, čili $\alpha = H_2(\alpha/e)$. Toto dosadíme do rov-
742 nice (4.1), upravíme podle definice entropické funkce a použijeme (4.3)

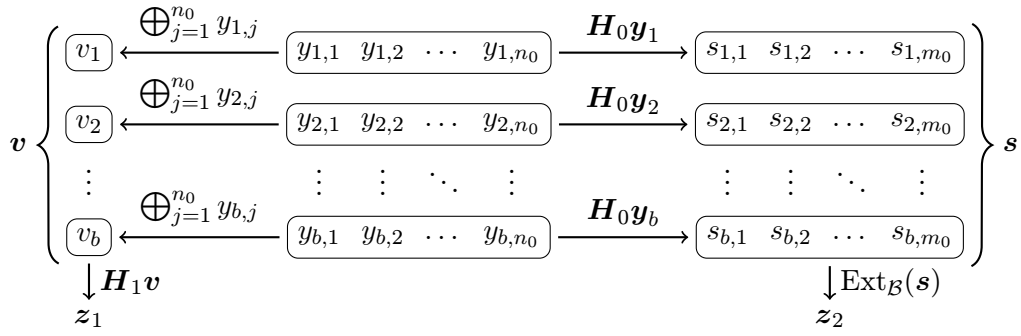
$$\alpha_{\pm 1} \approx H_2(\alpha/e) + \alpha/e = (\log_2 3)H_3(\alpha/e) = (\log_2 3)H_3(\alpha_{\pm 1}/e_{\pm 1}).$$

743 Odtud vyjádříme $e_{\pm 1} \approx \alpha_{\pm 1}/H_3^{-1}(\alpha_{\pm 1}/\log_2 3)$, což jsme měli dokázat. \square

744 **4.2.2. Stegosystém ZZW.** Začneme tím, že popíšeme algoritmus extrakce pro stegosys-
745 tém ZZW [12]. Stegoobjekt je rozdělen na b bloků velikosti $n_0 = 2^{m_0}$. Posloupnost hodnot
746 spjatých s i -tým blokem stegoobjektu značíme $\mathbf{y}_i = (y_{i,1}, \dots, y_{i,n_0})^T \in \mathbb{F}_2^{n_0}$. Extrakce
747 vložené zprávy je znázorněna na obrázku 4.1.

748 Stegosystém pracuje s maticemi \mathbf{H}_0 a \mathbf{H}_1 a s kódovou knihou \mathcal{B} .

- 749 • Matice \mathbf{H}_0 vzniká z paritní matice Hammingova $[2^{m_0} - 1, 2^{m_0} - 1 - m_0]_2$ kódu
750 přidáním nulového sloupce.
- 751 • Matice \mathbf{H}_1 je paritní matice nějakého $[b, b - m_1]_2$ kódu \mathcal{C} s průměrnou vzdáleností
752 od kódu r_a .



OBRÁZEK 4.1: ZZW extrakce

753 • Kódová kniha \mathcal{B} umožňuje vkládat m_2 bitů do nosiče délky bm_0 bitů, z nichž $r_a m_0$
754 je suchých.

755 Zpráva je rozdělena na dvě části $\mathbf{z}_1 \in \mathbb{F}_2^{m_1}$ a $\mathbf{z}_2 \in \mathbb{F}_2^{m_2}$. První část zprávy získáme tak, že
756 pro každé $i = 1, \dots, b$ spočteme $v_i = \bigoplus_{j=1}^{n_0} y_{i,j}$ a potom $\mathbf{z}_1 = \mathbf{H}_1(v_1, \dots, v_b)^T$. Druhou
757 část zprávy získáme tak, že pro každé $i = 1, \dots, b$ spočteme $\mathbf{s}_i = \mathbf{H}_0 \mathbf{y}_i$, všechna $\mathbf{s}_1, \dots, \mathbf{s}_b$
758 sřetězíme do jediného vektoru \mathbf{s} délky bm_0 a potom $\mathbf{z}_2 = \text{Ext}_{\mathcal{B}}(\mathbf{s})$.

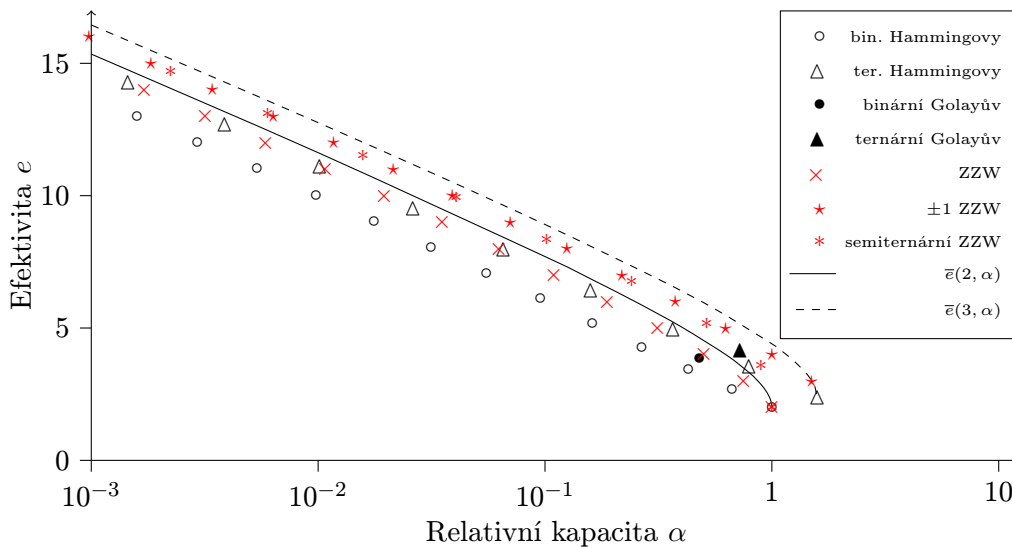
759 Nyní ukážeme, jak se takovýto stegoobjekt vytvoří. Vkládání dvojice zpráv $\mathbf{z}_1 \in \mathbb{F}_2^{m_1}$
760 a $\mathbf{z}_2 \in \mathbb{F}_2^{m_2}$ do nosiče délky $b2^{m_0}$ začneme tím, že nosič rozdělíme na b bloků velikosti $n_0 =$
761 2^{m_0} . Posloupnost hodnot spjatých s i -tým blokem nosiče značíme $\mathbf{x}_i = (x_{i,1}, \dots, x_{i,n_0})^T \in$
762 $\mathbb{F}_2^{n_0}$. Obdobně jako při extrakci spočteme vektor $\mathbf{u} \in \mathbb{F}_2^b$ tak, že $u_i = \bigoplus_{j=1}^{n_0} x_{i,j}$ pro
763 každé $i = 1, \dots, b$. Do tohoto vektoru vložíme první část zprávy, čímž dostaneme $\mathbf{v} =$
764 $\text{Emb}_{\mathbf{H}_1}(\mathbf{u}, \mathbf{z}_1)$. Tímto jsme získali návod na to, jak provést změny v nosiči. Jestliže $u_i = v_i$,
765 pak položíme $\mathbf{y}_i = \mathbf{x}_i$. Jestliže $u_i \neq v_i$, pak vektor \mathbf{y}_i získáme tak, že v \mathbf{x}_i provedeme
766 právě jednu změnu. Očekávaný počet změn vyvolaných vkládáním je tedy roven průměrné
767 vzdálenosti r_a od kódu \mathcal{C} . Máme-li provést jednu změnu ve vektoru \mathbf{x}_i , můžeme tak učinit
768 na libovolné pozici. Tato volnost je to, co nám dává prostor pro zakódování další zprávy \mathbf{z}_2 .

769 Jak víme, Hammingovy kódy umožňují vkládat informace provedením *nejvýše* jedné
770 změny v bloku nosiče. Matice \mathbf{H}_0 vznikla z paritní matice Hammingova kódu přidáním
771 jednoho nulového sloupce. Taková matice má potom vlastnost, že hodnotu syndromu $\mathbf{H}_0 \mathbf{x}_i$
772 lze upravit na libovolnou hodnotu provedením *právě* jedné změny ve vektoru \mathbf{x}_i . To zna-
773 mená, že když $u_i \neq v_i$, tak ve chvíli, kdy ve vektoru \mathbf{x}_i volíme pozici, na které provedeme
774 změnu, máme plnou kontrolu nad hodnotou syndromu $\mathbf{H}_0 \mathbf{y}_i$. Naopak když $u_i = v_i$, tak
775 nemáme žádnou kontrolu nad $\mathbf{H}_0 \mathbf{y}_i = \mathbf{H}_0 \mathbf{x}_i$. Pro každé $i = 1, \dots, b$ spočteme $\mathbf{r}_i = \mathbf{H}_0 \mathbf{x}_i$.
776 Když $u_i = v_i$, tak označíme složky vektoru \mathbf{r}_i jako mokré, v opačném případě jako su-
777 ché. Všechna $\mathbf{r}_1, \dots, \mathbf{r}_b$ sřetězíme do jediného vektoru \mathbf{r} . Vektor \mathbf{r} je nyní mokrý nosič
778 délky bm_0 a očekávaný počet suchých prvků je $r_a m_0$. Podle věty o mokrém nosiči lze do
779 takového nosiče vložit téměř $r_a m_0$ bitů informace.

780 Stegosystém ZZW je tedy schopen vložit téměř $m_1 + r_a m_0$ bitů do nosiče délky $b2^{m_0}$
781 provedením r_a změn. Proto máme relativní kapacitu a efektivitu

$$\alpha \approx \frac{m_1 + r_a m_0}{b2^{m_0}} \quad \text{a} \quad e \approx \frac{m_1 + r_a m_0}{r_a}.$$

782 Podívejme se na nejjednodušší případ, kdy za \mathcal{C} zvolíme triviální $[b, 0]_2$ kód, tj. \mathbf{H}_1 je
783 identická matice řádu b . Potom $\mathbf{z}_1 = \mathbf{v}$, $m_1 = b$ a $r_a = \frac{1}{2}b$. Čili máme $\alpha \approx 2^{-m_0}(1 + \frac{1}{2}m_0)$
784 a $e \approx 2 + m_0$ bez ohledu na počet bloků b . Na obrázku 4.2 máme graf efektivitu ZZW



OBRÁZEK 4.2: Efektivita ZZW vkládání v závislosti na relativní kapacitě ve srovnání s perfektními kódy a s horní mezí na spodní efektivitu pro $q = 2$ a 3 .

785 vkládání v závislosti na relativní kapacitě ve srovnání s perfektními kódy a s horní mezí na
786 spodní efektivitu pro $q = 2$ a 3 . Odstup ZZW vkládání od horní meze na spodní efektivitu
787 binárních kódů je méně než $0,6$.

788 Efektivitu vkládání můžeme ještě vylepšit tím, že navíc aplikujeme dvouúrovňové ± 1
789 vkládání. Odstup ± 1 ZZW vkládání od horní meze na spodní efektivitu ternárních kódů
790 je také méně než $0,6$.

791 Na závěr poukážeme na několik zobecnění tohoto schématu. Především si můžeme
792 všimnout, že \mathbf{u} funguje jako zcela samostatný nosič. Vkládání do tohoto vektoru tedy
793 nemusí být prováděno najednou jediným kódem \mathcal{C} , ale vektor \mathbf{u} lze rozdělit na bloky
794 menší délky a do těch vkládat zprávu \mathbf{z}_1 po částech. Dále si všimněme, že (upravená)
795 matice Hammingova kódu \mathbf{H}_0 nemusí být pro každý blok stejná. Délku Hammingova
796 kódu můžeme pro každý blok volit libovolně.

797 **Cvičení 4.3.** Vymyslete „semiternární“ variaci ZZW stegosystému, kde se namísto bi-
798 nárních Hammingových kódů použijí ternární Hammingovy kódy, avšak kód \mathcal{C} zůstává
799 binární. Ukažte, že relativní kapacita a efektivita takového schématu je

$$\alpha \approx \frac{2(m_1 + r_a m_0 \log_2 3)}{b(3^{m_0} + 1)} \quad \text{a} \quad e \approx \frac{m_1 + r_a m_0 \log_2 3}{r_a}.$$

800 **Cvičení 4.4.** Vymyslete ternární variaci ZZW stegosystému, kde všechny použité kódy
801 jsou ternární. Relativní kapacita a efektivita takového schématu by měla vyjít

$$\alpha \approx \frac{(m_1 + r_a m_0) \log_2 3}{b3^{m_0}} \quad \text{a} \quad e \approx \frac{(m_1 + r_a m_0) \log_2 3}{r_a}.$$

802 4.3. PSANÍ NA MOKRÝ PAPIR POMOCÍ MATIC

803 Stejně jako v kapitole 2 budeme značit $\mathbf{x}, \mathbf{y} \in \mathbb{F}_q^n$ posloupnost hodnot spjatých s blokem
804 nosiče a s blokem stegoobjektu a $\mathbf{z} \in \mathbb{F}_q^m$ budeme značit zprávu vkládanou do daného
805 bloku. Paritní matici samoopravného kódu typu $m \times n$ nad \mathbb{F}_q značíme jako obvykle \mathbf{H} .

806 Pro každý blok máme nyní navíc množinu indexů suchých složek, kterou značíme \mathcal{S} , a
807 množinu indexů mokrých složek $\mathcal{M} = \{1, \dots, n\} \setminus \mathcal{S}$. Počet suchých prvků budeme značit
808 $s := |\mathcal{S}|$. Pro každý vektor $\mathbf{x} \in \mathbb{F}_q^n$ definujeme vektor $\mathbf{x}_{\mathcal{S}} \in \mathbb{F}_q^s$, který vznikne zúžením
809 vektoru \mathbf{x} na složky indexované množinou \mathcal{S} . Podobně $\mathbf{H}_{\mathcal{S}}$ značíme podmatici matice \mathbf{H} ,
810 která vznikne zúžením matice \mathbf{H} na sloupce indexované množinou \mathcal{S} .

811 Extrakce zprávy \mathbf{z} ze stegoobjektu \mathbf{y} je shodná s definicí v kapitole 2, tj. $\mathbf{z} = \mathbf{H}\mathbf{y}$.

812 **Algoritmus 4.5** (maticové vkládání do mokrého nosiče).

vstup: nosič $\mathbf{x} \in \mathbb{F}_q^n$, zpráva $\mathbf{z} \in \mathbb{F}_q^m$, paritní matice $\mathbf{H} [n, n - m]_q$ kódu \mathcal{C} ,
813 množina mokrých indexů \mathcal{M}

výstup: stegoobjekt $\mathbf{y} \in \mathbb{F}_q^n$ takový, že $\mathbf{H}\mathbf{y} = \mathbf{z}$ a $\mathbf{y}_{\mathcal{M}} = \mathbf{x}_{\mathcal{M}}$

```

1   $\mathcal{S} := \{1, \dots, n\} \setminus \mathcal{M}$ 
2   $\mathbf{y}_{\mathcal{M}} := \mathbf{x}_{\mathcal{M}}$ 
3   $\mathbf{b} := \mathbf{z} - \mathbf{H}_{\mathcal{M}}\mathbf{x}_{\mathcal{M}}$ 
4  if  $(\mathbf{H}_{\mathcal{S}} \mid \mathbf{b})$  má řešení then
814 5     za  $\mathbf{y}_{\mathcal{S}}$  zvol libovolné řešení soustavy  $(\mathbf{H}_{\mathcal{S}} \mid \mathbf{b})$ 
6     return  $\mathbf{y}$ 
7  else
8     return fail
```

815 *Důkaz.* $\mathbf{H}\mathbf{y} = \mathbf{H}_{\mathcal{S}}\mathbf{y}_{\mathcal{S}} + \mathbf{H}_{\mathcal{M}}\mathbf{y}_{\mathcal{M}} = \mathbf{b} + \mathbf{H}_{\mathcal{M}}\mathbf{x}_{\mathcal{M}} = \mathbf{z}$. □

816 Nyní stojíme před problémem, jakým způsobem vybrat paritní matici \mathbf{H} , aby sou-
817 stava $(\mathbf{H}_{\mathcal{S}} \mid \mathbf{b})$ měla řešení a aby toto řešení bylo možné efektivně spočítat. Aby soustava
818 $(\mathbf{H}_{\mathcal{S}} \mid \mathbf{b})$ byla řešitelná pro každou pravou stranu, musí být hodnota matice $\mathbf{H}_{\mathcal{S}}$ rovna m .
819 Jinými slovy, její řádky musejí být lineárně nezávislé. Nutnou podmínkou je tedy, aby
820 $s \geq m$. Zdaleka nejlepší by bylo, kdyby matice \mathbf{H} měla vlastnost, že každá její podma-
821 tice $\mathbf{H}_{\mathcal{S}}$, kde $s \geq m$, má hodnotu m . Tuto vlastnost lze také formulovat tak, že každých
822 m sloupců matice \mathbf{H} tvoří lineárně nezávislou množinu. Kódy, jejichž paritní matice mají
823 tuto vlastnost, známe pod názvem *MDS kódy*. Bohužel pro $q = 2$ existují jen triviální MDS
824 kódy s parametry $[n, 1]_2$, $[n, n]_2$ nebo $[n, n - 1]_2$. Z netriviálních MDS kódů známe napří-
825 klad zobecněné Reed-Solomonovy kódy, ale jejich nevýhodou je, že délka těchto kódů je
826 omezena velikostí tělesa. Velikost tělesa bychom však chtěli udržovat minimální, abychom
827 minimalizovali velikost změn v nosiči. Naproti tomu délku kódu bychom chtěli spíše velkou,
828 jak plyne z následující úvahy.

829 Mějme pevně zvolený nosič a označme σ pravděpodobnost, že náhodně zvolený prvek
830 nosiče je suchý. Relativní počet symbolů vkládané zprávy označme $\mu = m/n$. V předchozím
831 odstavci jsme shledali, že aby algoritmus mohl fungovat, musíme volit μ menší než σ ,
832 což ostatně vyžaduje i věta o mokrému nosiči. Když rozdělíme nosič na velké bloky, pak
833 podle zákona velkých čísel bude počet suchých složek v bloku relativně blízko očekávané
834 hodnotě σn a můžeme očekávat, že bude větší než $\mu n = m$. Naproti tomu při rozdělení
835 nosiče na malé bloky stoupá hrozba, že do některého z bloků padne nedostatečný počet
836 suchých složek a algoritmus selže.

837 Jednou možností jak získat matici \mathbf{H} typu $\mu n \times n$ je sestavit ji náhodně. Potom
838 pravděpodobnost, že její libovolná podmatice typu $\mu n \times \sigma n$ má lineárně nezávislé řádky,
839 se s rostoucím n blíží jedné, za předpokladu, že $\mu < \sigma$.

840 **Lemma 4.6.** *Nechť m a n jsou přirozená čísla taková, že $m \leq n$. Potom pravděpodobnost,*
841 *že náhodně zvolená matice typu $m \times n$ nad tělesem \mathbb{F}_q má lineárně nezávislé řádky, je rovna*

$$\prod_{i=0}^{m-1} (1 - q^{i-n}).$$

842 *Důkaz.* Počet matic typu $m \times n$ nad \mathbb{F}_q s lineárně nezávislými řádky lze určit tak, že
 843 popíšeme jakým způsobem bychom je všechny sestrojili. Na první řádek můžeme zvolit
 844 kterýkoliv nenulový vektor, tj. máme $q^n - 1$ možností. Na $(i + 1)$ -ní řádek můžeme zvolit
 845 kterýkoliv vektor s výjimkou vektorů, které lze vyjádřit jako lineární kombinaci prvních
 846 i řádků, tj. máme $q^n - q^i$ možností. Celkem tedy máme $\prod_{i=0}^{m-1} (q^n - q^i)$ matic s lineárně
 847 nezávislými řádky. \square

848 **Lemma 4.7.** *Nechť $x \in [0, 1]$ a $m \geq 1$, pak $(1 - x)^m \geq 1 - mx$.*

849 *Důkaz.* Jestliže $mx \geq 1$, pak nerovnost zřejmě platí. Dále se tedy věnujme případu $mx < 1$.
 850 Funkce $f_m(x) = m \ln(1 - x) - \ln(1 - mx)$ má derivaci $f'_m(x) = -\frac{m}{1-x} + \frac{m}{1-mx}$ a ta je na
 851 intervalu $[0, 1/m)$ nezáporná. Funkce $f_m(x)$ je tedy neklesající na intervalu $[0, 1/m)$ a dále
 852 vidíme, že je na tomto intervalu spojitá a $f_m(0) = 0$. Odtud plyne, že $m \ln(1 - x) - \ln(1 -$
 853 $mx) \geq 0$ na $[0, 1/m)$. \square

854 **Tvrzení 4.8.** *Nechť $\mu, \delta, \sigma \in (0, 1]$ a $\mu < \sigma$. Potom pravděpodobnost, že náhodně zvolená
 855 matice typu $\lfloor \mu n \rfloor \times \lceil \sigma n \rceil$ nad tělesem \mathbb{F}_q má lineárně nezávislé řádky, je pro všechny
 856 dostatečně velké hodnoty n alespoň $1 - \delta$.*

857 *Důkaz.* Označme $m = \lfloor \mu n \rfloor$ a $s = \lceil \sigma n \rceil$. Pravděpodobnost, že náhodně zvolená matice
 858 typu $m \times s$ nad tělesem \mathbb{F}_q má lineárně nezávislé řádky, je

$$\prod_{i=0}^{m-1} (1 - q^{i-s}) > (1 - q^{m-s})^n \geq 1 - n q^{m-s} \geq 1 - n q^{(\mu-\sigma)n}.$$

859 \square

860 Bloky velké délky obecně činí z vkládání velice náročný problém. Časová složitost řešení
 861 soustavy rovnic pomocí Gaussovy eliminace je totiž kubická v délce bloku (pro pevné μ
 862 a σ). Pro efektivní řešení soustavy budeme muset zajistit, aby soustava měla nějaký vhodný
 863 tvar. Dobrá by byla například soustava v odstupňovaném tvaru, kterou bychom mohli
 864 vyřešit zpětnou substitucí v kvadratickém čase. Ještě lepší by byla soustava, jejíž řádky
 865 by byly nejenom odstupňované, ale navíc řídké, tj. až na pár složek nulové. Takovou
 866 soustavu vyřešíme v lineárním čase. Vzhledem k tomu, že soustava, kterou máme řešit
 867 vzniká v podstatě náhodným výběrem sloupců matice \mathbf{H} , můžeme zajistit její řídkost, ale
 868 není možné zajistit, aby byla vždy v odstupňovaném tvaru. Převod do odstupňovaného
 869 tvaru lze ovšem obstarat i jinak, než pomocí Gaussovy eliminace. V případě řídkých matic
 870 se to může podařit i pouhými permutacemi řádků a sloupců. Následující příklad ukazuje,
 871 jak na to.

872 **Příklad 4.9.** Pro jednoduchost předvedeme vkládání s pomocí paritní matice Hammin-
 873 gova $[7, 4]_2$ kódu. Uvažujme blok nosiče se třemi mokřými prvky (podtržené) 13, 12, 16, 17,
 874 16, 15, 14 a zprávu $\mathbf{z} = (1, 0, 1)^T$. Blok nosiče převedeme na vektor $\mathbf{x} = (\underline{1}, 0, 0, 1, 0, \underline{1}, 0)^T$.
 875 Máme tedy $\mathcal{S} = \{3, 4, 5, 7\}$, $\mathcal{M} = \{1, 2, 6\}$ a

$$\mathbf{H} = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}, \quad \mathbf{H}_{\mathcal{S}} = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix}, \quad \mathbf{H}_{\mathcal{M}} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix}.$$

876 Dále máme $\mathbf{x}_{\mathcal{S}} = (0, 1, 0, 0)^T$ a $\mathbf{x}_{\mathcal{M}} = (1, 0, 1)^T$. Spočteme $\mathbf{b} = \mathbf{z} - \mathbf{H}_{\mathcal{M}} \mathbf{x}_{\mathcal{M}} = (0, 1, 0)^T$ a
 877 najdeme řešení $\mathbf{y}_{\mathcal{S}}$ soustavy

878

$$(\mathbf{H}_S | \mathbf{b}) = \left(\begin{array}{cccc|c} & y_3 & y_4 & y_5 & y_7 & \\ 0 & 1 & 1 & 1 & & 0 \\ 1 & 0 & 0 & 1 & & 1 \\ 1 & 0 & 1 & 1 & & 0 \end{array} \right).$$

879 Nejdříve najdeme sloupec, který obsahuje jen jednu jedničku, a vyměníme ho s prvním
880 sloupcem.

881

$$\left(\begin{array}{cccc|c} & y_4 & y_3 & y_5 & y_7 & \\ 1 & 0 & 1 & 1 & & 0 \\ 0 & 1 & 0 & 1 & & 1 \\ 0 & 1 & 1 & 1 & & 0 \end{array} \right)$$

882 V dalším kroku najdeme sloupec, který obsahuje jen jednu jedničku v posledních dvou
883 složkách, a vyměníme ho s druhým sloupcem. Potom přeuspořádáme řádky tak, aby jed-
884 nička padla na diagonálu.

885

$$\left(\begin{array}{cccc|c} & y_4 & y_5 & y_3 & y_7 & \\ 1 & 1 & 0 & 1 & & 0 \\ 0 & 0 & 1 & 1 & & 1 \\ 0 & 1 & 1 & 1 & & 0 \end{array} \right) \rightarrow \left(\begin{array}{cccc|c} & y_4 & y_5 & y_3 & y_7 & \\ 1 & 1 & 0 & 1 & & 0 \\ 0 & 1 & 1 & 1 & & 0 \\ 0 & 0 & 1 & 1 & & 1 \end{array} \right)$$

886 Tím už máme soustavu v odstupňovaném tvaru. Hodnotu y_7 můžeme zvolit libovolně.
887 Abychom minimalizovali počet změn, vezmeme $y_7 = x_7 = 0$. Zpětnou substitucí dořešíme
888 soustavu: $y_3 = 1$, $y_5 = 1$ a $y_4 = 1$. Máme tedy $\mathbf{y} = (1, 0, \mathbf{1}, 1, \mathbf{1}, 1, 0)^T$ a blok stegoobjektu
889 13, 12, **17**, 17, **17**, 15, 14.

890 Snadno nahlédneme, že kdyby množina indexů mokřých složek byla $\mathcal{M} = \{1, 2, 4\}$, pak
891 už by soustava nebyla řešitelná pomocí řádkových a sloupcových permutací.

892 Matici \mathbf{H}_S obecně převádíme do odstupňovaného tvaru tak, že v levém horním rohu
893 postupně vytváříme horní trojúhelníkovou matici s jedničkami na diagonále. V k -tém
894 kroku najdeme sloupec, který ve své spodní části obsahuje pouze jednu jedničku. Tuto
895 jedničku pak přemístíme na diagonálu permutacemi řádků a sloupců.

896

$$\left(\begin{array}{cccc|cccc} & 1 & \dots & k-1 & k & & j & \\ 1 & * & * & & * & * & * & * \\ 0 & 1 & * & & * & * & * & * \\ 0 & 0 & 1 & & * & * & * & * \\ k & 0 & 0 & 0 & * & * & 0 & * \\ i & 0 & 0 & 0 & * & * & \mathbf{1} & * \\ 0 & 0 & 0 & & * & * & 0 & * \end{array} \right) \rightarrow \left(\begin{array}{cccc|cccc} & 1 & \dots & k-1 & j & & k & \\ 1 & * & * & * & * & * & * & * \\ 0 & 1 & * & * & * & * & * & * \\ 0 & 0 & 1 & * & * & * & * & * \\ 0 & 0 & 0 & \mathbf{1} & * & * & * & * \\ k & 0 & 0 & 0 & 0 & * & * & * \\ 0 & 0 & 0 & 0 & * & * & * & * \end{array} \right)$$

897 **Algoritmus 4.10** (řešení soustavy permutacemi řádků a sloupců).

898 **vstup:** soustava $(\mathbf{A} | \mathbf{b})$, kde \mathbf{A} je typu $m \times s$ nad \mathbb{F}_q , vektor $\mathbf{u} \in \mathbb{F}_q^s$
899 **výstup:** řešení soustavy, které se na alespoň $s - m$ pozicích shoduje s \mathbf{u}

```

1   $\pi := \text{id} \in S_s$ 
2  for  $k = 1, \dots, m$  do
3      v matici  $\mathbf{A}$  najdi  $j$ -tý sloupec takový, že  $w(a_{k,j}, \dots, a_{m,j}) = 1$ 
4      if  $j$  neexistuje then
5          return fail
6      buď  $i \geq k$  takové, že  $a_{i,j} \neq 0$ 
7      swap( $\mathbf{A}_{i*}, \mathbf{A}_{k*}$ )
8      swap( $b_i, b_k$ )
9      swap( $\mathbf{A}_{*j}, \mathbf{A}_{*k}$ )

```

```

10      $\pi := \pi \circ (j, k)$ 
11   for  $k = m, \dots, 1$  do
12      $u_{\pi(k)} := a_{kk}^{-1}(b_k - \sum_{i=k+1}^s a_{ki} u_{\pi(i)})$ 
13   return  $\mathbf{u}$ 

```

900 **Definice.** Necht m je přirozené číslo, $\delta > 0$ a $c > 0$. Označme $R = c \cdot \ln(m/\delta)\sqrt{m}$,

$$\rho(i) = \begin{cases} \frac{1}{m} & \text{pro } i = 1, \\ \frac{1}{i(i-1)} & \text{pro } i = 2, \dots, m, \\ 0 & \text{pro } i > m, \end{cases} \quad \text{a} \quad \tau(i) = \begin{cases} \frac{R}{m} \frac{1}{i} & \text{pro } i = 1, \dots, m/R - 1, \\ \frac{R}{m} \ln(R/\delta) & \text{pro } i = m/R, \\ 0 & \text{pro } i > m/R. \end{cases}$$

901 Řekneme, že diskrétní náhodná veličina X má *robustní solitonové rozdělení* s parametry
902 (m, δ, c) , jestliže

$$\Pr[X = i] = \frac{\rho(i) + \tau(i)}{\sum_{j=1}^m \rho(j) + \tau(j)}.$$

903 Předchozí definice byla poprvé představena v článku [10], který pojednává o tzv. LT
904 kódech. Tyto samoopravné kódy jsou určené pro binární výmazový kanál a lze je po-
905 psat tak, že Hammingovy váhy sloupců jejich generující matice mají robustní solitonové
906 rozdělení. Z uvedeného článku plyne následující věta.

907 **Věta 4.11.** *Máme-li soustavu rovnic nad tělesem \mathbb{F}_q s alespoň $s \approx m + c\sqrt{m}(\ln(m/\delta))^2$
908 sloupci takovými, že jejich Hammingovy váhy mají robustní solitonové rozdělení s para-
909 metry (m, δ, c) , pak pravděpodobnost selhání algoritmu 4.10 je nejvýše δ .*

910 Předchozí věta nám dává návod na sestavení matice \mathbf{H} . Zvolíme velikost bloku n ,
911 horní mez δ na pravděpodobnost selhání algoritmu a parametr c , který je vhodné volit
912 řádově okolo 10^{-1} . V závislosti na aplikaci máme obvykle dán buď relativní počet suchých
913 prvků v nosiči σ , nebo relativní délku zprávy μ , a podle věty 4.11 určíme druhý z nich
914 tak, aby

$$\sigma > \mu + c \frac{\sqrt{\mu n}}{n} \left(\ln \frac{\mu n}{\delta} \right)^2. \quad (4.4)$$

915 Matici \mathbf{H} typu $\mu n \times n$ nad tělesem \mathbb{F}_q sestojíme náhodně tak, aby Hammingovy váhy
916 jejích sloupců měly robustní solitonové rozdělení s parametry $(\mu n, \delta, c)$. Při vkládání se ze
917 sloupců této matice vybere podmatice \mathbf{H}_S velikosti $\mu n \times \sigma n$. Hammingovy váhy sloupců
918 matice \mathbf{H}_S mají totéž rozdělení jako Hammingovy váhy sloupců matice \mathbf{H} .

919 Z rovnice (4.4) zároveň vidíme, že pro pevně zvolené σ , δ a c se odstup μ od σ s ros-
920 toucím n blíží nule, což je další důkaz věty o mokrému nosiči.

921 **Tvrzení 4.12.** *Jestliže Hammingovy váhy sloupců matice \mathbf{H} typu $m \times n$ mají robustní
922 solitonové rozdělení s parametry (m, δ, c) , pak časová složitost maticového ukládání do
923 mokrého nosiče je $O(n \log(m/\delta))$.*

924 *Důkaz.* Začneme tím, že spočítáme horní odhad na očekávanou Hammingovu váhu libo-
925 volného sloupce matice \mathbf{H} . Především si všimněme, že

$$\sum_{i=1}^{\infty} \rho(i) = \frac{1}{m} + \sum_{i=2}^m \frac{1}{i(i-1)} = \frac{1}{m} + \sum_{i=2}^m \frac{1}{i-1} - \frac{1}{i} = 1.$$

Očekávaná hodnota náhodné veličiny s robustním solitonovým rozdělením je

$$\begin{aligned} \sum_{i=1}^{\infty} i \cdot \frac{\rho(i) + \tau(i)}{\sum_{j=1}^m \rho(j) + \tau(j)} &\leq \sum_{i=1}^{\infty} i(\rho(i) + \tau(i)) = \frac{1}{m} + \sum_{i=2}^m \frac{1}{i-1} + \sum_{i=1}^{m/R-1} \frac{R}{m} + \ln \frac{R}{\delta} \\ &\leq \ln(m) + 1 + 1 + \ln \frac{c \cdot \ln(m/\delta) \sqrt{m}}{\delta} = O\left(\log \frac{m}{\delta}\right) \end{aligned}$$

926 V předposledním kroku jsme využili odhadu $\sum_{i=1}^m 1/i \leq \log(m) + 1$.

927 Předpokládejme, že matice \mathbf{H} je uložena po řádcích v řídké reprezentaci. Očekávaná
928 váha řádku matice \mathbf{H}_S je $O\left(\frac{s}{m} \log \frac{m}{\delta}\right)$.

929 Složitost algoritmu 4.5 spočívá ve výpočtu vektoru $\mathbf{b} = \mathbf{z} - \mathbf{H}_{\mathcal{M}} \mathbf{x}_{\mathcal{M}}$, sestavení matice
930 \mathbf{H}_S a řešení soustavy $(\mathbf{H}_S | \mathbf{b})$. Výpočet \mathbf{b} a sestavení \mathbf{H}_S lze provést současně a vyžaduje
931 jeden průchod všech prvků matice \mathbf{H} . Počet nenulových prvků v \mathbf{H} je celkem $O(n \log \frac{m}{\delta})$.
932 Zbývá určit složitost algoritmu 4.10.

933 Nejdříve spočítáme složitost jednoho průchodu cyklem, který začíná na řádce 2. Cyklus
934 začíná nalezením sloupce, jehož spodní část má váhu 1. Počet nenulových hodnot ve spodní
935 části sloupce nemusíme počítat při každém průchodu cyklem, ale stačí mít tyto údaje
936 uložené a při každém přesunu řádku do horní části matice je pouze aktualizovat. Nalezení
937 sloupce má konstantní složitost a aktualizace vah spodních částí sloupců je má složitost
938 $O\left(\frac{s}{m} \log \frac{m}{\delta}\right)$. Operace swap na řádcích 7 a 9 algoritmu není třeba provádět fyzicky, stačí
939 aby se permutace prováděly například na ukazatelích, a tedy v konstantním čase. Celková
940 složitost m průchodů cyklem je $O\left(s \log \frac{m}{\delta}\right)$.

941 Složitost jednoho průchodu cyklem, který začíná na řádce 11 je $O\left(\frac{s}{m} \log \frac{m}{\delta}\right)$. Celková
942 složitost m průchodů tímto cyklem je tedy opět $O\left(s \log \frac{m}{\delta}\right)$.

943 Složitost algoritmu 4.5 je tedy $O\left(s \log \frac{m}{\delta}\right) = \left(n \log \frac{m}{\delta}\right)$. □

944 Nyní máme návod na sestrojení matice \mathbf{H} a také efektivní algoritmy vkládání a ex-
945 trakce. Otázkou zůstává, jakým způsobem si strany vymění matici \mathbf{H} . Jednou možností
946 samozřejmě je, aby matice byla předem dohodnutá. Nevýhodou takového postupu je, že
947 jsme vázáni na určité předem určené hodnoty σ a μ . Taková matice nebude použitelná
948 pro nosiče s relativním počtem suchých prvků menším než σ . Naopak pro nosiče s relativ-
949 ním počtem suchých prvků větším než σ nám pevně zvolená matice nedovolí plně využít
950 kapacitu nosiče.

951 Další možností je předem dohodnout určité parametry, jako například velikost bloku n
952 a parametry c a δ , ale délku zprávy m a matici \mathbf{H} zvolit až ve chvíli, kdy máme kon-
953 krétní nosič a zprávu. V takovém případě musíme v nosiči vyhradit nějaký malý prostor,
954 kam uložíme informaci o délce zprávy. Paritní matice se potom sestojí náhodně, přičemž
955 jako inicializační hodnota generátoru náhodných čísel může sloužit právě délka zprávy m .
956 Vzhledem k tomu, že obě strany sdílejí všechny parametry pro sestrojení matice, můžou
957 si ji každá samostatně vygenerovat. Tento postup má jednu velkou výhodu. Stane-li se, že
958 pro některý blok nosiče algoritmus 4.10 selže, můžeme situaci vyřešit tak, že celý proces
959 začneme od začátku s jinou maticí \mathbf{H} . To se dá zařídit tak, že zprávu doplníme nějakým
960 bezvýznamným znakem, čímž změníme její délku a tím i inicializační hodnotu generátoru
961 náhodných matic.

962 5. VKLÁDÁNÍ POMOCÍ VITERBIHO ALGORITMU

963 V této kapitole vysvětlíme, jakým způsobem lze ve steganografii využít teorii konvolučních
964 kódů, zejména popíšeme použití Viterbiho algoritmu k minimalizaci distorze vyvolané

965 vkládáním. Významnou motivací je, že do Viterbiho algoritmu lze přirozeně zakomponovat
 966 zohlednění vah jednotlivých změn v nosiči. S myšlenkou použití konvolučních kódů ve
 967 steganografii přišli Filler, Judas a Fridrich v článku [4]. Jejich metodu vyložíme v řeči
 968 předmětu Automaty a konvoluční kódy přednášeného na MFF. Někteří čtenáři mohou
 969 upřednostnit ryze maticový popis metody v původním článku [4].

970 Stejně jako v kapitole 2 rozdělujeme nosič na bloky velikosti n a prvkům nosiče
 971 přiřazujeme hodnoty z konečného tělesa. Na rozdíl od kapitoly 2 nebudeme pracovat s
 972 jednotlivými bloky samostatně, ale sestrojíme z nich posloupnost vektorů $\{\mathbf{x}_i\}_{i=0}^{\ell-1}$, kde
 973 $\mathbf{x}_i = (x_i^{(1)}, \dots, x_i^{(n)})^T \in \mathbb{F}_q^n$. Pro jednoduchost budeme opět hovořit přímo o $\{\mathbf{x}_i\}_{i=0}^{\ell-1}$ jako
 974 o nosiči a o $\{\mathbf{y}_i\}_{i=0}^{\ell-1}$ jako o stegoobjektu. Zpráva je posloupnost vektorů $\{\mathbf{z}_i\}_{i=0}^{\ell-1}$, kde
 975 $\mathbf{z}_i \in \mathbb{F}_q^m$.

976 Relativní kapacita nosiče je stejná jako v případě maticového vkládání $\alpha = m(\log_2 q)/n$.
 977 V tomto textu se budeme soustředit na případ $m = 1$. Relativní kapacita nosiče je potom
 978 sice omezena na hodnoty tvaru $\alpha = (\log_2 q)/n$, to ale nepředstavuje problém, protože ve
 979 steganografii máme tendenci cílit na hodnoty blízké nule. Na zprávu je pak přirozenější
 980 pohlížet jako na posloupnost skalárů $\{z_i\}_{i=0}^{\ell-1}$, než jako na posloupnost jednosložkových vek-
 981 torů. Zobecnění pro $m > 1$ je jednoduché, viz poznámka 5.2, nezdá se však být přínosné,
 982 protože neúměrně navyšuje složitost Viterbiho algoritmu.

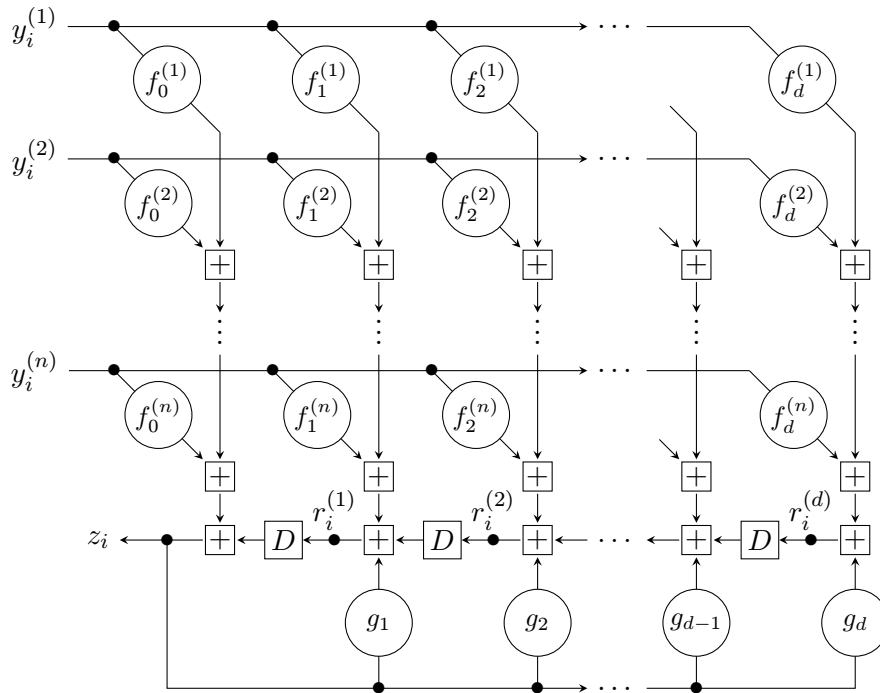
5.1. KONVOLUČNÍ EXTRAKCE

983
 984 Extrakce zprávy se provádí konvolučním překladačem. Kdybychom překladač implemen-
 985 tovali pomocí kodéru v kontrolorově normální formě, potřebovali bychom n paměťových
 986 registrů, protože v každém kroku přijímá kodér n vstupních hodnot. Překladač lze také
 987 implementovat kodérem v pozorovatelově normální formě, který je popsán na obrázku 5.1
 988 a využívá pouze $m = 1$ paměťový registr délky d . Proměnnou, která odpovídá j -té buňce
 989 registru na konci i -tého kroku, značíme $r_i^{(j)}$. D značí operátor zpoždění. V čase $i = 0$ je
 990 celý paměťový registr vynulovaný, tj. $r_0^{(j)} = 0$ pro $1 \leq j \leq d$. Hodnoty $f_j^{(k)} \in \mathbb{F}_q$ a $g_j \in \mathbb{F}_q$,
 991 kde $0 \leq j \leq d$ a $1 \leq k \leq n$, jsou předem stanovené konstanty, přičemž konstanta g_0 , která
 992 se v obrázku ani neobjevuje, je vždy volena rovna 0. Prochází-li hrana zakroužkovanou
 993 konstantou, znamená to, že se příslušná hodnota touto konstantou násobí.

994 Pro referenci popíšeme extrakční algoritmus také pomocí pseudokódu.

995 **Algoritmus 5.1** (konvoluční extrakce).

vstup: stegoobjekt $\{\mathbf{y}_i\}_{i=0}^{\ell-1}$ z \mathbb{F}_q^n ,
 996 parametry $f_j^{(k)} \in \mathbb{F}_q$ a $g_j \in \mathbb{F}_q$, kde $0 \leq j \leq d$ a $1 \leq k \leq n$
výstup: zpráva $\{z_i\}_{i=0}^{\ell-1}$
 1 $(s_0, \dots, s_d) := (0, \dots, 0)$
 2 **for** $i = 0, \dots, \ell - 1$ **do**
 3 **for** $j = 1, \dots, n$ **do**
 4 $(s_0, \dots, s_d) := (s_0, \dots, s_d) + y_i^{(j)}(f_0^{(j)}, \dots, f_d^{(j)})$
 997 $z_i := s_0$
 5 $(s_0, \dots, s_d) := (s_0, \dots, s_d) + s_0(g_0, \dots, g_d)$
 6 $(s_0, \dots, s_d) := (s_1, \dots, s_d, 0)$
 7 **return** $\{z_i\}_{i=0}^{\ell-1}$
 8



OBRÁZEK 5.1: Konvoluční extrakce

998

5.2. SOUVISLOST S KONVOLUČNÍMI KÓDY

Abychom porozuměli souvislosti s konvolučními kódy, popíšeme fungování extrakčního algoritmu pomocí rovnic. V i -tém kroku algoritmu platí

$$z_i = r_{i-1}^{(1)} + \sum_{k=1}^n f_0^{(k)} y_i^{(k)},$$

$$r_i^{(j)} = r_{i-1}^{(j+1)} + g_j z_i + \sum_{k=1}^n f_j^{(k)} y_i^{(k)}, \quad \text{pro } 1 \leq j < d,$$

$$r_i^{(d)} = g_d z_i + \sum_{k=1}^n f_d^{(k)} y_i^{(k)}.$$

999 Úpravou těchto rovnic se můžeme zbavit registrových proměnných a vyjádřit z_i :

$$z_i = \sum_{j=1}^d g_j z_{i-j} + \sum_{k=1}^n \sum_{j=0}^d f_j^{(k)} y_{i-j}^{(k)}.$$

1000 Uvážíme-li, že $g_0 = 0$, pak se tato rovnost dá zapsat pomocí D -transformací jako

$$z(D) = g(D)z(D) + \sum_{k=1}^n f^{(k)}(D) y^{(k)}(D),$$

1001 což po úpravě dává

$$z(D) = \sum_{k=1}^n y^{(k)}(D) \frac{f^{(k)}(D)}{1 - g(D)} = \frac{1}{1 - g(D)} \mathbf{f}(D) \mathbf{y}(D),$$

1002 kde $\mathbf{f}(D) = (f^{(1)}(D), \dots, f^{(n)}(D))$ je matice typu $1 \times n$ nad tělesem racionálních funkcí
 1003 $\mathbb{F}_q(D)$ a $\mathbf{y}(D) = (y^{(1)}(D), \dots, y^{(n)}(D))^T$ je vektor polynomů z $\mathbb{F}_q[D]$. Extrakční algoritmus
 1004 tedy skutečně je konvoluční překladač.

1005 *Poznámka 5.2.* Kdybychom chtěli sestrojít kodér v pozorovatelově normální formě pro
 1006 matici \mathbf{f} s více řádky, stačilo by pro každý řádek sestrojít kodér popsany na obrázku 5.1.
 1007 Každý z m kodérů by měl stejný vstup a kodéry by tedy běžely paralelně vedle sebe. Jejich
 1008 výstupy by se pak poskládaly do posloupnosti m -složkových vektorů $\{z_i\}_{i=0}^{\ell-1}$.

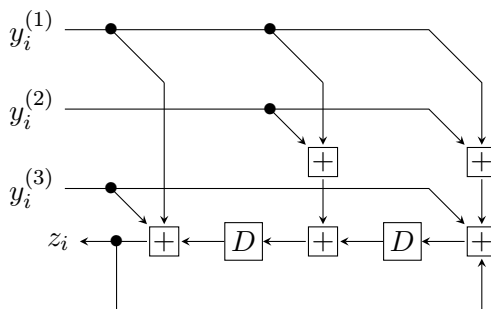
1009 Nevýhodou většího počtu kodérů je větší celková délka paměťových registrů a tedy
 1010 exponenciální nárůst počtu stavů výsledného automatu.

1011

5.3. VKLÁDÁNÍ

1012 Extrakční algoritmus nejdříve popíšeme jako konečný překladač. Pro tento překladač se-
 1013 strojíme trelážový diagram, což je graf, který popisuje, jak se vyvíjejí stavy překladače v
 1014 závislosti na vstupu. Cesty v trelážovém grafu obecně odpovídají všem možným vstupům
 1015 překladače. My ovšem sestavíme náš graf tak, aby obsahoval pouze ty cesty (vstupy), je-
 1016 jichž výstupem je vkládaná zpráva $\{z_i\}_{i=0}^{\ell-1}$. Tyto cesty jsou kandidátky na stegoobjekt.
 1017 Viterbiho algoritmus potom najde cestu s nejmenší distorzí.

1018 Tento postup si nejlépe ukážeme na příkladu. Mějme extrakční automat popsany na
 obrázku 5.2. Tento automat odpovídá konečnému překladači na obrázku 5.3. Stav zde



OBRÁZEK 5.2: Konvoluční extrakce s parametry $\mathbf{f}(D) = (1 + D + D^2, D + D^2, 1 + D^2)$ a $g(D) = D^2$ nad \mathbb{F}_2

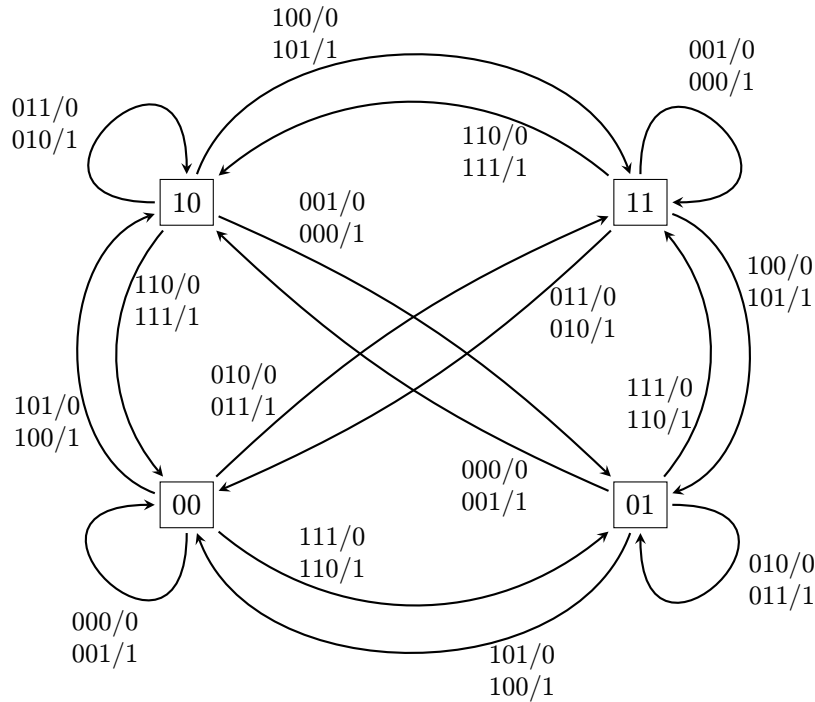
1019

1020 zapisujeme jako dvojici $r_i^{(1)}r_i^{(2)}$. Fungování tohoto překladače si popíšeme na příkladu
 1021 hrany, která vede ze stavu 10 do stavu 11 a je označena dvěma instrukcemi 100/0 a 101/1.
 1022 To znamená, že je-li překladač ve stavu 10 a vstupem je 100 nebo 101, pak překladač
 1023 přejde do stavu 11 a v případě prvního vstupu vypíše 0, ve druhém případě vypíše 1.

1024 Z překladače sestrojíme dva trelážové moduly. Jeden bude obsahovat přechody, které
 1025 mají výstup 0, druhý ty, které mají výstup 1.

1026 V závislosti na tom, jakou zprávu chceme vložit, pospojujeme příslušné moduly do
 1027 treláže. V treláži pak najdeme pomocí Viterbiho algoritmu cestu, která se nejméně liší od
 1028 nosiče, anebo obecněji řečeno cestu, která má nejmenší váhu vzhledem k nějaké distorzní
 1029 funkci.

1030 Všimněme si v čem se tato treláž liší od té, se kterou se setkáváme při dekodování
 1031 konvolučních kódů. Jednak zde nevyžadujeme, aby překladač skončil v nulovém stavu.
 1032 Hrany označujeme vstupními hodnotami překladače místo výstupních hodnot. Naším cílem
 1033 je totiž upravovat vstupní hodnoty, abychom dosáhli požadovaný výstup. Naproti tomu v
 1034 případě dekodéru konvolučních kódů nám jde o to upravovat poškozený výstup, abychom
 1035 dosáhli platný výstup.



OBRÁZEK 5.3: Konečný překladač odpovídající kodéru na obrázku 5.2

1036 Kdybychom spustili Viterbiho algoritmus na takto sestrojené treláži, byla by jeho
 1037 časová složitost $O(q^{d+n-1}\ell n)$ operací porovnání. Rozvinutím modulů se dá vylepšit časová
 1038 složitost na $O(q^{d+2}\ell n)$ vektorových operací. Rozvinutí spočívá v tom, že každý přechod
 1039 extrakčního automatu rozdělíme na $n + 1$ podkroků. Tyto podkroky odpovídají operaci
 1040 na řádku 4 v algoritmu 5.1, která se provede n -krát v každém přechodu, a operacím na
 1041 řádku 6 a 7, které se provedou jedenkrát v každém přechodu. Stavový registr je třeba
 1042 rozšířit o jednu buňku, čímž se počet stavů znásobí q -krát. Příklad rozvinutých modulů je
 1043 uveden na obrázcích 5.6 a 5.7. Rozdíl mezi nimi se projevuje pouze v posledním podkroku.

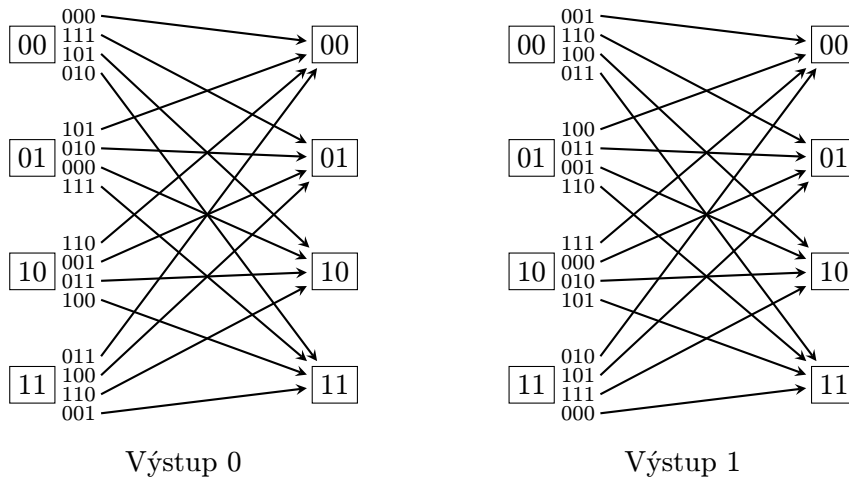
1044 Jak už jsme zmiňovali, Viterbiho algoritmus zohledňuje váhy jednotlivých změn v
 1045 nosiči. Ke každému prvku nosiče $x_i^{(j)}$ přiřazujeme distorzní funkci $\rho_i^{(j)} : \mathbb{F}_q \rightarrow \mathbb{R}$. Hodnota
 1046 $\rho_i^{(j)}(a)$ udává, jak moc by přispělo nastavení $y_i^{(j)} = a$ ve stegoobjektu k celkové distorzi
 1047 vyvolané vkládáním. Chceme-li použít jednoduchou metriku, která pouze sleduje počet
 1048 změn v nosiči, definujeme

$$\rho_i^{(j)}(a) = \begin{cases} 0, & \text{jestliže } a = x_i^{(j)}, \\ 1, & \text{jestliže } a \neq x_i^{(j)}. \end{cases}$$

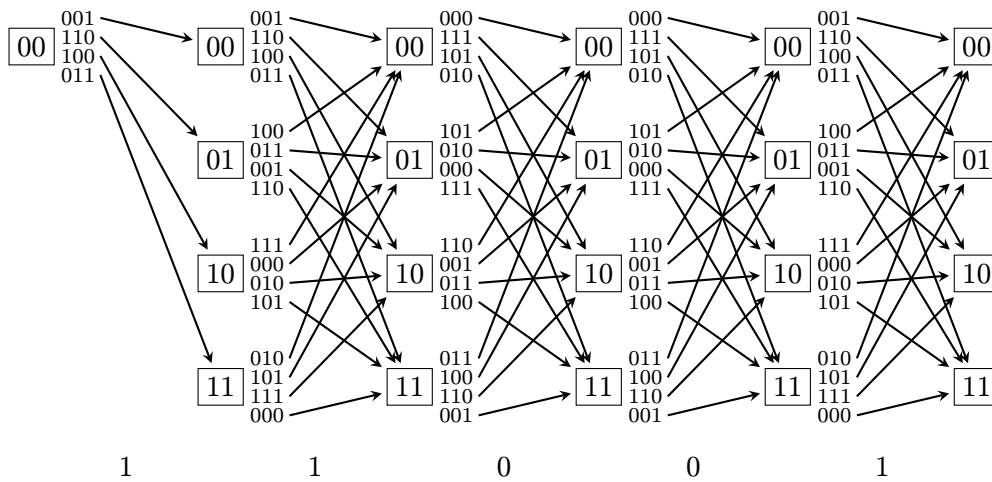
1049 Vhodnou volbou distorzní funkce můžeme tuto metodu použít i k psaní na mokřý papír:

$$\rho_i^{(j)}(a) = \begin{cases} \infty, & \text{jestliže } a \neq x_i^{(j)} \text{ a prvek na } j\text{-té pozici v } i\text{-tém bloku je mokřý,} \\ 0 & \text{jinak.} \end{cases}$$

1050 V první části Viterbiho algoritmu se prochází treláž zleva doprava, tj. pro $i = 0, \dots, \ell -$
 1051 1 a pro $j = 1, \dots, n$. Pro každé i a j máme až q^{d+1} stavů s , ve kterých se může automat v
 1052 daném okamžiku nacházet. Pro každý z těchto stavů se spočítá váha nejlehčí cesty, která
 1053 do něho vede, a do $path_i^{(j)}[s]$ se zaznamená poslední hrana této cesty. Jakmile se dojde na
 1054 konec treláže, vybere se stav, do kterého vede nejlehčí cesta a tato se zpětným průchodem



OBRÁZEK 5.4: Trelážové moduly sestavené z konečného překladače na obrázku 5.3



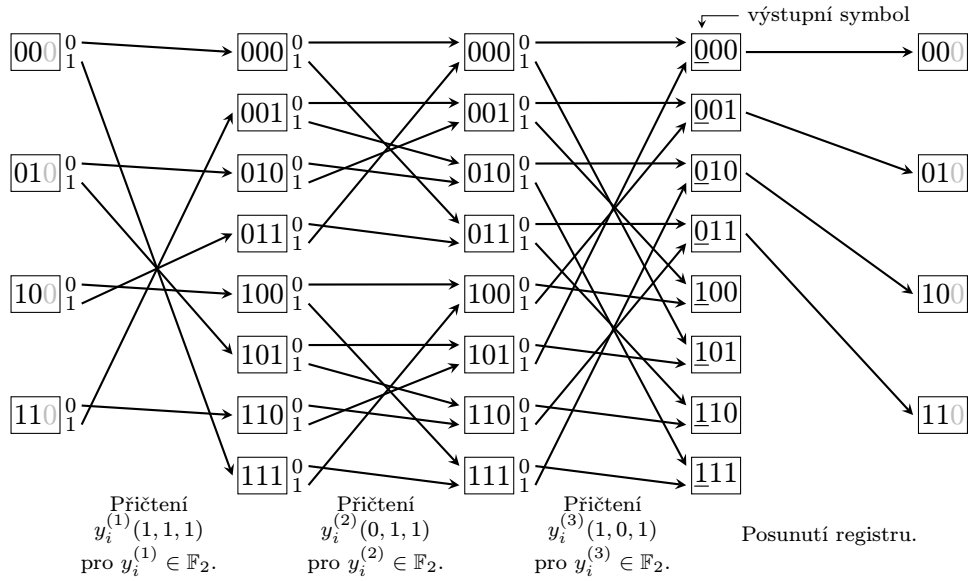
OBRÁZEK 5.5: Treláž pro vložení zprávy 11001 sestavená z modulů na obrázku 5.4

1055 zrekonstruuje. Algoritmus vyžaduje paměť k uložení $O(q^{d+1}\ell n)$ prvků tělesa \mathbb{F}_q a čas k
 1056 provedení $O(q^{d+2}\ell n)$ operací na registru. Celý postup je popsán v algoritmu 5.3. Tento
 1057 popis je velmi obecný. Pro běžné použití by měla stačit jeho specializace pro $q = 2$ bez
 1058 zpětné vazby, kterou lze implementovat pomocí operací XOR. Tato specializovaná varianta
 1059 algoritmu je k nalezení v článku [4].

1060 **Algoritmus 5.3** (vkládání pomocí Viterbiho algoritmu).

vstup: distorzní funkce prvků nosiče $\{\rho_i\}_{i=0}^{\ell-1}$, zpráva $\{z_i\}_{i=0}^{\ell-1}$ nad \mathbb{F}_q ,
 1061 parametry extrakčního algoritmu $f(D)$ a $g(D)$

výstup: stegoobjekt $\{y_i\}_{i=0}^{\ell-1}$



OBRÁZEK 5.6: Rozvinutý trelážový modul z obrázku 5.4 s výstupem 0.

— Inicializace.

```

1 for  $s \in \mathbb{F}_q^{d+1} \setminus \{\mathbf{0}\}$  do
2    $weight[s] := \infty$ 
3  $weight[\mathbf{0}] := 0$ 

```

— Dopředný průchod treláží.

```

4 for  $i = 0, \dots, \ell - 1$  do
5   for  $j = 1, \dots, n$  do
6     for  $s \in \mathbb{F}_q^{d+1}$  do
7        $path_i^{(j)}[s] := \arg \min_{a \in \mathbb{F}_q} weight[s - a(f_0^{(j)}, \dots, f_d^{(j)})] + \rho_i^{(j)}(a)$ 
8        $weight'[s] := \min_{a \in \mathbb{F}_q} weight[s - a(f_0^{(j)}, \dots, f_d^{(j)})] + \rho_i^{(j)}(a)$ 
9        $weight := weight'$ 
10    for  $(s_0, \dots, s_{d-1}) \in \mathbb{F}_q^d$  do
11       $weight'[(s_0, \dots, s_{d-1}, 0)] := weight[(z_i, s_0, \dots, s_{d-1}) - z_i(g^{(1)}, \dots, g^{(n)})]$ 
12    for  $s_d \in \mathbb{F}_q \setminus \{0\}$  do
13       $weight'[(s_0, \dots, s_d)] := \infty$ 
14     $weight := weight'$ 

```

— Volba stavu, ve kterém končí nejlehčí cesta.

```

15  $(s_0, \dots, s_d) := \arg \min_{s' \in \mathbb{F}_q^{d+1}} weight[s']$ 

```

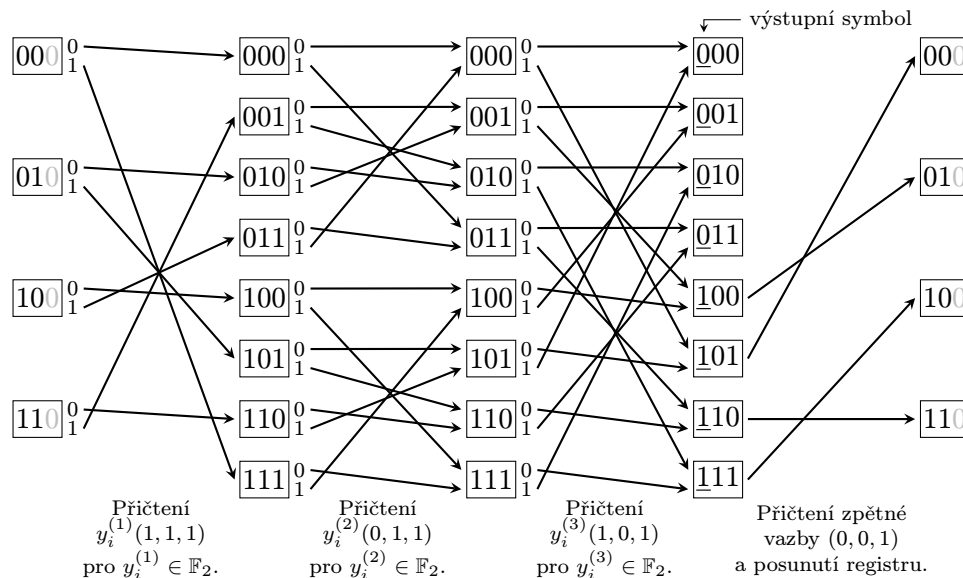
— Rekonstrukce nejlehčí cesty zpětným průchodem treláže.

```

16 for  $i = \ell - 1, \dots, 0$  do
17    $(s_0, s_1, \dots, s_d) := (z_i, s_0, \dots, s_{d-1}) - z_i(g_1, \dots, g_n)$ 
18   for  $j = n, \dots, 1$  do
19      $y_i^{(j)} := path_i^{(j)}[(s_0, \dots, s_d)]$ 
20      $(s_0, \dots, s_d) = (s_0, \dots, s_d) - y_i^{(j)}(f_0^{(j)}, \dots, f_d^{(j)})$ 
21 return  $\{y_i\}_{i=0}^{\ell-1}$ 

```

1063 Stegosystém založený na konvolučním překladači uvedeném na obrázku 5.2 dosahuje
 1064 efektivity $e = 3,87$, přičemž $\alpha = 1/3$. Toto velmi dobře odpovídá efektivitě Hammingo-
 1065 vých kódů v dané oblasti. Nicméně použijeme-li větší registr, např. $d = 10$, pak existují



OBRÁZEK 5.7: Rozvinutý trelážový modul z obrázku 5.4 s výstupem 1.

1066 konvoluční překladače pro $\alpha = 1/3$, s nimiž lze dosáhnout efektivity 4,94.

1067 Experimenty naznačují, že přítomnost zpětné vazby $g(D)$ nemá měřitelný vliv na efek-
1068 tivitu vkládání.

1069

LITERATURA

- 1070 [1] Berlekamp, E. R.; McEliece, R. J.; van Tilborg, H. C. A.: On the inherent intractability of certain coding problems. *Information Theory, IEEE Transactions on*, ročník 24,
1071 č. 3, May 1978: s. 384–386, ISSN 0018-9448, doi:10.1109/TIT.1978.1055873.
1072 URL <http://dx.doi.org/10.1109/TIT.1978.1055873>
1073
- 1074 [2] Cohen, G.; Honkala, I.; Litsyn, S.; aj.: *Covering codes, North-Holland Mathematical*
1075 *Library*, ročník 54. Elsevier, 1997, ISBN 9780080530079.
- 1076 [3] Crandall, R.: Some notes on steganography, December 1998, posted on steganography
1077 mailing list.
1078 URL <http://os.inf.tu-dresden.de/westfeld/crandall.pdf>
- 1079 [4] Filler, T.; Judas, J.; Fridrich, J.: Minimizing Additive Distortion in Steganography
1080 Using Syndrome-Trellis Codes. *Information Forensics and Security, IEEE Transac-*
1081 *tions on*, ročník 6, č. 3, Sept 2011: s. 920–935, ISSN 1556-6013, doi:10.1109/TIFS.2011.
1082 2134094.
1083 URL <http://dx.doi.org/10.1109/TIFS.2011.2134094>
- 1084 [5] Fridrich, J.: *Steganography in digital media: Principles, algorithms, and applications*.
1085 Cambridge University Press, 2010.
- 1086 [6] Fridrich, J.; Goljan, M.; Soukal, D.: Perturbed quantization steganography. *Mul-*
1087 *timedia Systems*, ročník 11, č. 2, 2005: s. 98–107, ISSN 0942-4962, doi:10.1007/
1088 s00530-005-0194-3.
1089 URL <http://dx.doi.org/10.1007/s00530-005-0194-3>

- 1090 [7] Fridrich, J.; Lisoněk, P.; Soukal, D.: On Steganographic Embedding Efficiency. In
1091 *Information Hiding, Lecture Notes in Computer Science*, ročník 4437, editace J. L.
1092 Camenisch; C. S. Collberg; N. F. Johnson; P. Sallee, Springer Berlin Heidelberg, 2007,
1093 ISBN 978-3-540-74123-7, s. 282–296, doi:10.1007/978-3-540-74124-4_19.
1094 URL http://dx.doi.org/10.1007/978-3-540-74124-4_19
- 1095 [8] Kim, Y.; Duric, Z.; Richards, D.: Modified Matrix Encoding Technique for Mini-
1096 mal Distortion Steganography. In *Information Hiding, Lecture Notes in Computer*
1097 *Science*, ročník 4437, editace J. Camenisch; C. Collberg; N. Johnson; P. Sallee,
1098 Springer Berlin Heidelberg, 2007, ISBN 978-3-540-74123-7, s. 314–327, doi:10.1007/
1099 978-3-540-74124-4_21.
1100 URL http://dx.doi.org/10.1007/978-3-540-74124-4_21
- 1101 [9] Li, X.; Zeng, T.; Yang, B.: Improvement of the embedding efficiency of LSB matching
1102 by sum and difference covering set. In *Multimedia and Expo, 2008 IEEE International*
1103 *Conference on*, June 2008, s. 209–212, doi:10.1109/ICME.2008.4607408.
1104 URL <http://dx.doi.org/10.1109/ICME.2008.4607408>
- 1105 [10] Luby, M.: LT codes. In *Foundations of Computer Science, 2002. Proceedings. The*
1106 *43rd Annual IEEE Symposium on*, 2002, ISSN 0272-5428, s. 271–280, doi:10.1109/
1107 SFCS.2002.1181950.
1108 URL <http://dx.doi.org/10.1109/SFCS.2002.1181950>
- 1109 [11] Zhang, W.; Zhang, X.; Wang, S.: A Double Layered “Plus-Minus One” Data Embed-
1110 ding Scheme. *Signal Processing Letters, IEEE*, ročník 14, č. 11, Nov 2007: s. 848–851,
1111 ISSN 1070-9908, doi:10.1109/LSP.2007.903255.
1112 URL <http://dx.doi.org/10.1109/LSP.2007.903255>
- 1113 [12] Zhang, W.; Zhang, X.; Wang, S.: Maximizing Steganographic Embedding Effici-
1114 ency by Combining Hamming Codes and Wet Paper Codes. In *Information Hiding,*
1115 *Lecture Notes in Computer Science*, ročník 5284, editace K. Solanki; K. Sullivan;
1116 U. Madhow, Springer Berlin Heidelberg, 2008, ISBN 978-3-540-88960-1, s. 60–71, doi:
1117 10.1007/978-3-540-88961-8_5.
1118 URL http://dx.doi.org/10.1007/978-3-540-88961-8_5