STEGANOGRAFIE A DIGITÁLNÍ MÉDIA

ANDREW KOZLÍK

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.

1. STEGANOGRAFIE V PALETOVÝCH OBRÁZCÍCH

1.1. Úvod

Na úvod připomeňme pár základních poznatků o reprezentaci barev. Barvy jsou reprezentovány jako uspořádané trojice $(r, g, b) \in \{0, 1, \ldots, 255\}^3$, 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 barvonosné složky. Transformace mezi reprezentacemi RGB a CIELAB je nelineární.

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 $I = \{0, 1, \ldots, |C| - 1\}$. Paleta je zpravidla uložena v souboru jako tabulka velkosti $|C| \times 3$ 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 přečísluje matice indexů.

Steganografii v paletových obrázcích lze provádět dvěma základními způsoby.

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

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

2. Vkládání do matice indexů.

   (a) Paletu setřídíme podle jasu. Bity zprávy vkládáme do nejnižších bitů jednotlivých indexů v matici. Uspořádání palety podle jasu zajistí, že změna nejnižšího bitu v indexu příliš neovlivní jas pixelu. Tento postup používá algoritmus EzStego. Jeho nevýhodou je, že změna nejnižšího bitu v indexu může vést k zásadní změně odstínu pixelu.

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

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

1.3. Vkládání s optimálním přiřazením parity

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

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

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

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

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

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

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

Než přistoupíme k algoritmu pro sestrojení optimálního přiřazení parity, uvedeme příslušné algoritmy vkládání a extrakce.

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

vstup: nosič $x = (x_1, \ldots, x_n) \in I^n$, zpráva $z = (z_1, \ldots, z_m) \in \{0,1\}^m$, klíč $\pi \in S_n$, optimální přiřazení parity (p, f)

výstup: stegoobjekt $y = (y_1, \ldots, y_n) \in I^n$

```
1  for i = 1, ..., n do
2      if i > m or p(x_π(i)) = z_i then
3          y_π(i) := x_π(i)
4      else
5          y_π(i) := f(x_π(i))
6  return y
```

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

vstup: stegoobjekt $y = (y_1, \ldots, y_n) \in I^n$, klíč $\pi \in S_n$, délka zprávy m, optimální přiřazení parity (p, f)

výstup: zpráva $z = (z_1, \ldots, z_m) \in \{0,1\}^m$

```
1  for i = 1, ..., m do
2      z_i := p(y_π(i))
3  return z
```

Algoritmus vkládání i algoritmus extrakce oba pracují s optimálním přiřazením parity (p, f) pro daný obrázek. Měli bychom zdůraznit, že pár (p, f) se v rámci komunikace neposílá, ale odesílatel i příjemce si ho spočítají samostatně ze znalosti palety. Aby stegosystém byl funkční, je nezbytné, aby obě strany dospěly ke stejnému přiřazení parity. Zároveň bychom uvítali, kdyby se tak stalo i v případě, že při přenosu dojde k přeuspořádání palety. Následující algoritmus pro sestrojení optimálního přiřazení parity tuto vlastnost splňuje.

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

vstup: paleta bez duplicit $C = \{ c_i \mid i \in I \}$

výstup: optimální přiřazení parity (p, f) pro C, nezávislé na uspořádání palety C

```
1  E := { (‖c_i − c_j‖, c_i, c_j) | i, j ∈ I, i ≠ j }
```

```
2  while E ≠ ∅ do
3     (d, c_i, c_j) := min_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 \ (ℝ × {c_i, c_j} × C)
10    else
11       p(i) := 1 − p(j)
12       f(i) := j
13       E := E \ (ℝ × {c_i} × C)
14 return (p, f)
```

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

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

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

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

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

2. STEGANOGRAFIE POMOCÍ MATICOVÉHO VKLÁDÁNÍ

2.1. Úvod

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

bloku vložili bity zprávy např. do prvních dvou prvků a třetí prvek bychom nechali nedotčený. Jinou možností je rozložit bity zprávy do nejnižších bitů celého bloku takovým způsobem, aby příjemce z každého bloku stegoobjektu (y_1, y_2, y_3) extrahoval zprávu (z_1, z_2) 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 probíhá tak, že vyšetříme, která z následujících čtyř variant nastává a provedeme příslušnou 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

Pro lepší náhled můžeme tuto tabulku porovnat s odpovídající tabulkou pro standardní 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

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

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

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

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

2.2. Maticové vkládání pomocí Hammingových kódů

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

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

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

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

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

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

V praxi bychom vkládání pomocí Hammingových kódů realizovali následující způsobem. Na vstupu dostaneme nosič a zprávu. Najdeme největší celé číslo m takové že

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

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

Nyní přejděme ke q-árním Hammingovým kódů. Jak už jsem zmínili v úvodu, prvky nosiče asociujeme s prvky tělesa $\mathbb{F}_q$ pomocí operace $x \mapsto x \bmod q$. Jestliže q není prvočíslo, pak hodnotám $\{1, 2, \ldots, q-1\}$ přiřadíme prvky $\mathbb{F}_q \setminus \{0\}$ libovolným způsobem. Než budeme pokračovat, všimněme si, že i bez použití maticového vkládání lze tímto přístupem dosáhnout lepší efektivity vkládání než u klasického vkládání do nejnižšího bitu. Pro $q=3$ můžeme do každého prvku nosiče vložit jeden ternární symbol zprávy 0, 1, nebo 2 tak, že prvek necháme jak je, anebo jeho hodnotu zvýšíme či snížíme o 1, aby zbytek po dělení 3 byl roven vkládanému symbolu zprávy. Do každého prvku tak vkládáme $\log_2 3 \approx 1{,}585$ 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 efektivity vkládání $e = 3\,(\log_2 3)/2 \approx 2{,}377$. Pro vyšší hodnoty q už ovšem efektivita této metody klesá.

Algoritmus 2.1 (vkládání pomocí Hammingova kódu)**.**
vstup: nosič $\boldsymbol{x} \in \mathbb{F}_q^n$, zpráva $\boldsymbol{z} \in \mathbb{F}_q^m$,
paritní matice $\boldsymbol{H}$ Hammingova $[n, n-m]_q$ kódu
výstup: stegoobjekt $\boldsymbol{y} \in \mathbb{F}_q^n$ takový, že $\boldsymbol{Hy} = \boldsymbol{z}$

```
if Hx = z then
    return x
else
    najdi j-tý sloupec matice H a a ∈ F_q takové, že z − Hx = aH_{*j}
    return x + ae_j
```

Důkaz. $\boldsymbol{Hy} = \boldsymbol{Hx} + a\boldsymbol{He}_j = \boldsymbol{Hx} + a\boldsymbol{H}_{*j} = \boldsymbol{z}$. □

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

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

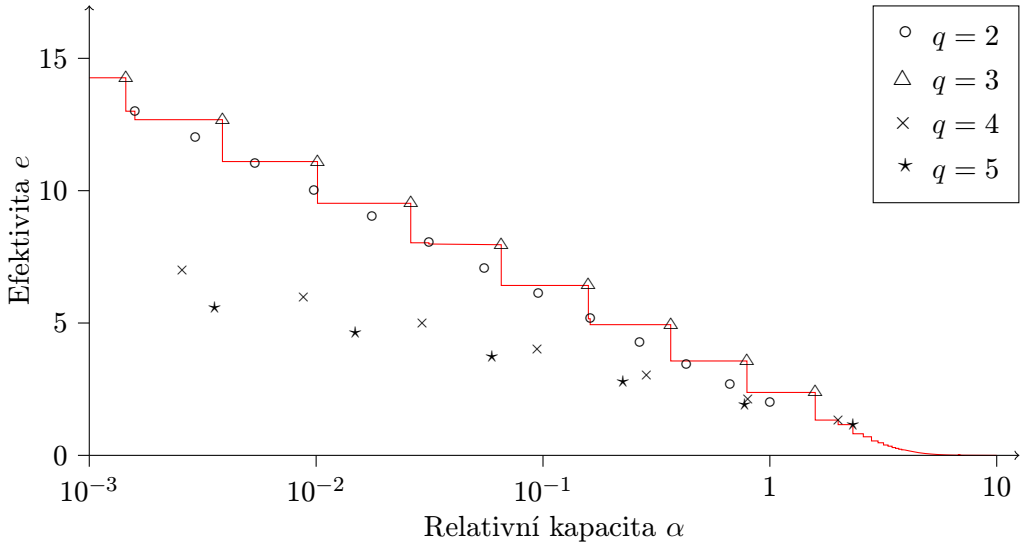
Do níže uvedeného bloku nosiče budeme vkládat zprávu $\boldsymbol{z} = (1, 2, 0)^{\mathrm{T}} \in \mathbb{F}_3^3$. Složky vektoru $\boldsymbol{x}$ získáme z prvků nosiče jako zbytek po dělení třemi.

Blok nosiče:	20	21	19	19	18	16	19	20	24	23	20	20	19
$\boldsymbol{x} = (\,$	2	0	1	1	0	1	1	2	0	2	2	2	$1\,)^{\mathrm{T}}$

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

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

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.

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

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

Relativní kapacita vkládání pomocí q-árních Hammingových kódů je

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

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

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

Na obrázku 2.1 máme graf efektivity 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, 5 a několik nejnižších hodnot m. Červená čára vyznačuje nejvyšší možnou efektivitu dosažitelnou pomocí Hammingových kódů pro dané α. Ve většině případů se této maximální efektivity dosahuje ternárními kódy. Výjimečně může být výhodnější použít binární kód a pro $\alpha > \log_2 3$ v podstatě i víceární kódy. Pro $\alpha \geq 1$ se však ve skutečnosti už nejedná o maticové vkládání v pravém slova smyslu, jde totiž o vkládání pomocí $[1, 0]_q$ kódu.

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

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

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

2.3. Maticové vkládání obecně

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

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

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

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

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

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

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

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

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

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

Algoritmus 2.3 (maticové vkládání)**.**
vstup: nosič $\boldsymbol{x} \in \mathbb{F}_q^n$, zpráva $\boldsymbol{z} \in \mathbb{F}_q^{n-k}$, paritní matice $\boldsymbol{H}$ $[n,k]_q$ kódu $\mathcal{C}$, minimum-distance dekodér D kódu $\mathcal{C}$
výstup: $\mathrm{Emb}_{\boldsymbol{H}}(\boldsymbol{x},\boldsymbol{z})$
1 zvol $\boldsymbol{u} \in \mathbb{F}_q^n$ takové, že $\boldsymbol{H}\boldsymbol{u} = \boldsymbol{z}$
2 **return** $D(\boldsymbol{x}-\boldsymbol{u})+\boldsymbol{u}$

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

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

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

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

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

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

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

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

$$r_{\mathrm{c}} := \max_{\boldsymbol{u}\in\mathbb{F}_q^n} d_{\mathrm{H}}(\boldsymbol{u},\mathcal{C})$$

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

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

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

$$\max_{\boldsymbol{x},\boldsymbol{z}} d_{\mathrm{H}}(\boldsymbol{x},\mathrm{Emb}_{\boldsymbol{H}}(\boldsymbol{x},\boldsymbol{z})) = r_{\mathrm{c}} \quad a \quad \mathop{\mathrm{E}}_{\boldsymbol{x},\boldsymbol{z}}[d_{\mathrm{H}}(\boldsymbol{x},\mathrm{Emb}_{\boldsymbol{H}}(\boldsymbol{x},\boldsymbol{z}))] = r_{\mathrm{a}},$$

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

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

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

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

$$\max_{\boldsymbol{z}\in\mathbb{F}_q^{n-k}} d_{\mathrm{H}}(\boldsymbol{x},\mathrm{Emb}_{\boldsymbol{H}}(\boldsymbol{x},\boldsymbol{z})) = \max_{\boldsymbol{z}\in\mathbb{F}_q^{n-k}} w(\mathbf{e}(\boldsymbol{z}\ -\boldsymbol{H}\boldsymbol{x})) =$$

$$\max_{\boldsymbol{s}\in\mathbb{F}_q^{n-k}} w(\mathbf{e}(\boldsymbol{s})) = \max_{\substack{\boldsymbol{s}\in\mathbb{F}_q^{n-k}\\ \boldsymbol{u}\in\mathcal{C}(\boldsymbol{s})}} d_{\mathrm{H}}(\boldsymbol{u},\mathcal{C}) = \max_{\boldsymbol{u}\in\mathbb{F}_q^n} d_{\mathrm{H}}(\boldsymbol{u},\mathcal{C}) = r_{\mathrm{c}}.$$

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

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

$$\mathop{\mathrm{E}}_{\boldsymbol{x},\boldsymbol{z}}[d_{\mathrm{H}}(\boldsymbol{x}, \mathrm{Emb}_{\boldsymbol{H}}(\boldsymbol{x},\boldsymbol{z}))] = \mathop{\mathrm{E}}_{\boldsymbol{s}}[w(\mathbf{e}(\boldsymbol{s}))] = \frac{1}{q^{n-k}} \sum_{\boldsymbol{s}\in\mathbb{F}_q^{n-k}} w(\mathbf{e}(\boldsymbol{s})) =$$

$$\frac{1}{q^n} \sum_{\boldsymbol{s}\in\mathbb{F}_q^{n-k}} q^k w(\mathbf{e}(\boldsymbol{s})) = \frac{1}{q^n} \sum_{\boldsymbol{s}\in\mathbb{F}_q^{n-k}} \sum_{\boldsymbol{u}\in\mathcal{C}(\boldsymbol{s})} d_{\mathrm{H}}(\boldsymbol{u},\mathcal{C}) = \frac{1}{q^n} \sum_{\boldsymbol{u}\in\mathbb{F}_q^n} d_{\mathrm{H}}(\boldsymbol{u},\mathcal{C}) = r_{\mathrm{a}}.$$

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

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

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

V případě těles $\mathbb{F}_q$, $q \geq 4$, je situace o něco složitější. Připomeňme, že pro q liché přicházejí v úvahu změny $\delta \in \Delta_q := \{\frac{-q+1}{2}, \ldots, -1, 0, 1, \ldots, \frac{q-1}{2}\}$ a pro q sudé $\delta \in \Delta_q := \{\frac{-q}{2}+1, \ldots, -1, 0, 1, \ldots, \frac{q}{2}\}$. Předpokládáme, že každá nenulová změna je stejně pravděpodobná. Distorze potom vychází $r_{\mathrm{a}}(q-1)^{-1} \sum_{\delta\in\Delta_q} \delta^2$, kde r_{a} je průměrná vzdálenost od příslušného kódu.

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

Definice. Pro $[n, k]_q$ kód s pokrývacím poloměrem r_{c} definujeme *spodní efektivitu* vkládání

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

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

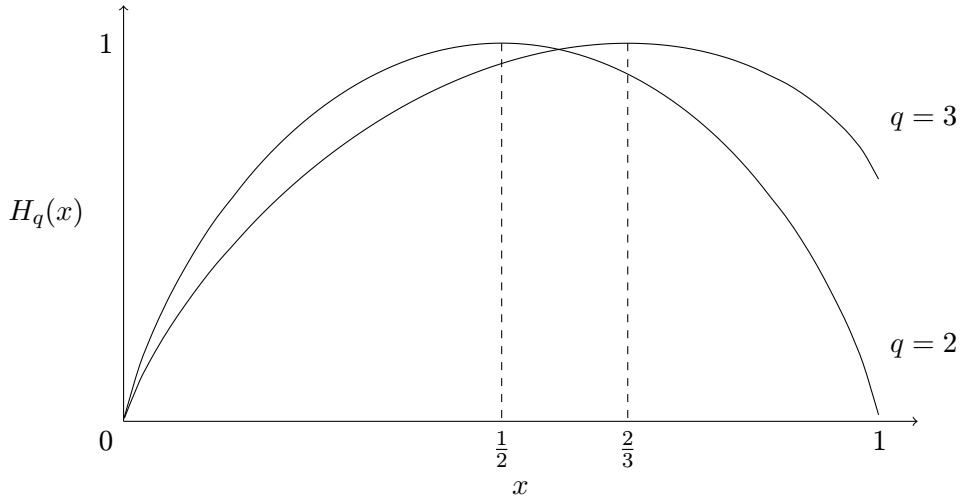
2.4. Spodní efektivita vkládání a velikost tělesa

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

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

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

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.

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

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

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

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

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

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

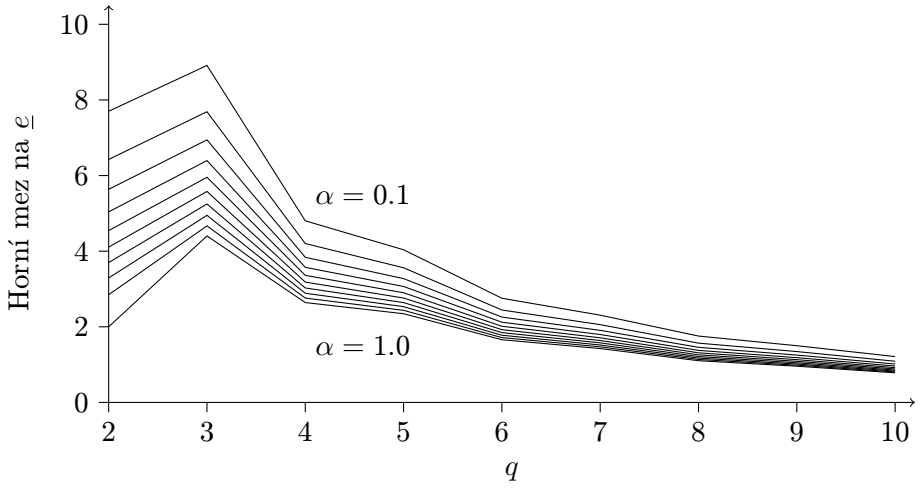
Dále vyjádříme n^n pomocí binomického rozvoje a získáme spodní odhad.

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

V posledním kroku jsme využili předpokladu, že $\frac{r}{(n-r)(q-1)} \leq 1$. Po dosazení nerovnosti 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. □

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

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



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

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

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

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

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

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

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

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

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

tedy pro nějaký kód byla spodní efektivita kódu dobrou aproximací jeho skutečné efektivity, je třeba, aby hodnoty r_{c} a r_{a} tohoto kódu měly k sobě blízko. Následující věta potvrzuje, že alespoň v případě „většiny“ binárních kódů tomu tak skutečně je.

Věta 2.9 ([7], Theorem 4)**.** *Nechť $0 < \alpha < 1$ a $\varepsilon > 0$. Podíl všech $[n, (1-\alpha)n]_2$ kódů, pro něž platí $|r_{\mathrm{c}} - r_{\mathrm{a}}|/n \leq \varepsilon$, konverguje k 1 pro $n \to \infty$.*

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

2.5. Perfektní kódy a horní mez na efektivitu vkládání

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

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

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

Důkaz. Označme $\mathcal{D}$ multimnožinu $\{\, w(\boldsymbol{c}-\boldsymbol{u}) \mid \boldsymbol{u} \in \mathbb{F}_q^n,\ \boldsymbol{c} \in \mathcal{C} \,\}$. Každá hodnota $i \in \{0,1,\ldots,n\}$ má v $\mathcal{D}$ četnost $q^k(q-1)^i\binom{n}{i}$. Našim cílem je získat spodní odhad výrazu $\sum_{\boldsymbol{u}\in\mathbb{F}_q^n} \min_{\boldsymbol{c}\in\mathcal{C}} w(\boldsymbol{c}-\boldsymbol{u})$. Tento výraz vyjadřuje součet určité q^n-prvkové podmultimnožiny multimnožiny $\mathcal{D}$, a můžeme ho proto zespodu odhadnout součtem q^n nejmenších hodnot 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 součtem nejvýše q^n nejmenších hodnot z $\mathcal{D}$

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

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

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

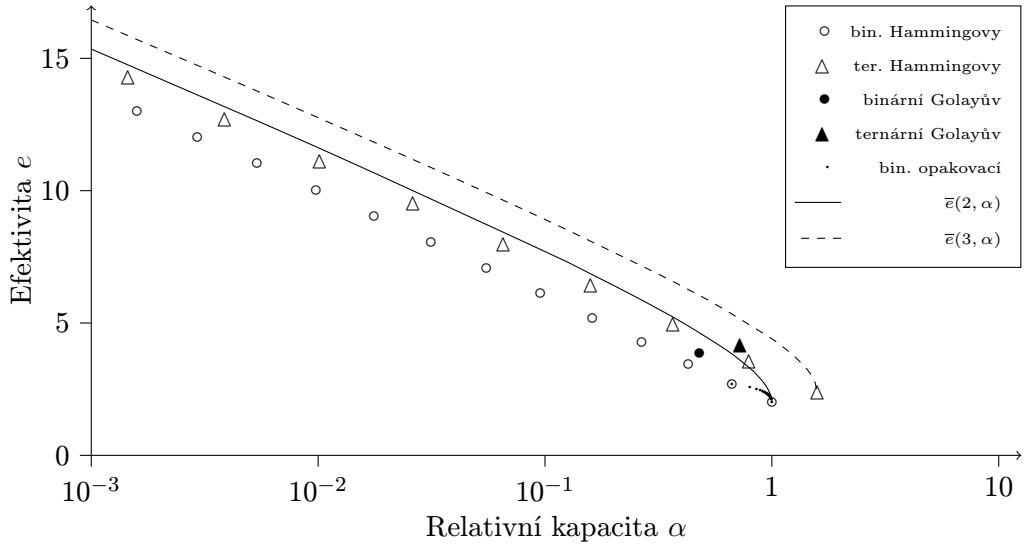
Připomeňme, že mezi perfektní kódy patří všechny úplné kódy, jednobodové kódy a binární opakovací kódy liché délky. Tyto jsou známy jako triviální perfektní kódy. Každý netriviální perfektní kód má parametry Hammingova kódu, Golayova $[23, 12, 7]_2$ kódu, anebo Golayova $[11, 6, 5]_3$ kódu.

Spojením předchozího tvrzení s důsledkem 2.5 získáme následující.

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

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

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.

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

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

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

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

3. SOUČTOVĚ A ROZDÍLOVĚ POKRÝVACÍ MNOŽINY

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

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

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 $\mathrm{Ext}(y_1, y_2) = (y_1 + 2y_2) \bmod 4$. 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 $\mathrm{Ext}(x_1, x_2) = (x_1 + 2x_2) \bmod 4 = 3$, 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 $\mathcal{A} = \{a_1, \ldots, a_n\}$ je posloupnost po dvou různých hodnot z G. Jestliže pro každé $g \in G$ existují $s_1, \ldots, s_n \in \{-1, 0, 1\}$ 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

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

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 poloměrem r_c lze sestrojit SDCS s parametry $(n, r_c, \mathbb{F}_3^{n-k})$ jednoduše tak, že za $\mathcal{A}$ zvolíme sloupce jeho paritní matice.

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

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

Algoritmus 3.1 (SDCS vkládání)**.**

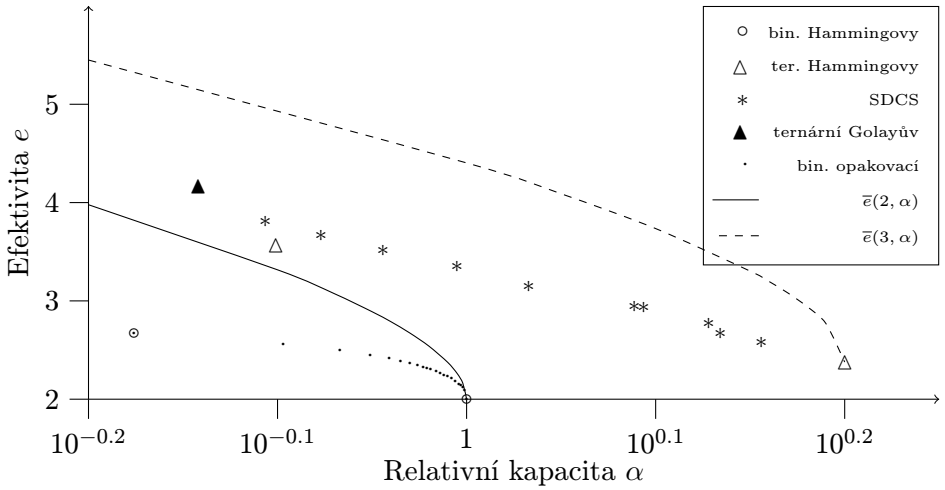
vstup: SDCS $\mathcal{A} = (a_1, \ldots, a_n) \in G^n$, blok nosiče $(x_1, \ldots, x_n) \in \mathbb{Z}^n$, zpráva $z \in G$
výstup: blok stegoobjektu $(y_1, \ldots, y_n) \in \mathbb{Z}^n$ takový, že $\mathrm{Ext}_{\mathcal{A}}(y_1, \ldots, y_n) = z$

1 $g := z - \sum_{i=1}^{n} x_i a_i$
2 najdi $s_1, \ldots, s_n$ 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** $(x_1 + s_1, \ldots, x_n + s_n)$

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

(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.

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

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

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

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

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

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

Z tohoto můžeme odvodit obdobu věty 2.7.

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

Důkaz. Jestliže $r/n \geq \frac{2}{3}$ pak tvrzení věty platí triviálně. Dále tedy předpokládejme, že $r/n < \frac{2}{3}$. 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 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 $\log_3|G| = n\alpha/\log_2 3$. □

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

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

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

4. PSANÍ NA MOKRÝ PAPÍR

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

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

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

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

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

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

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

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

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

Pro $n \to \infty$ 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 menší než 1. Z toho plyne, že pravděpodobnost, že $\mathcal{P}_{\boldsymbol{z}}$ neobsahuje vhodnou n-tici, se s rostoucím n blíží nule. □

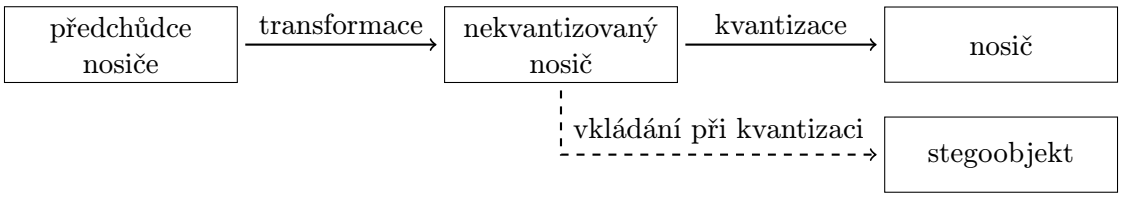
Pro algoritmy vkládání a extrakce popsané v důkazu předchozí věty můžeme použít označení $\boldsymbol{y} = \mathrm{Emb}_{\mathcal{B}}(\boldsymbol{x}, \mathcal{S}, \boldsymbol{z})$ a $\boldsymbol{z} = \mathrm{Ext}_{\mathcal{B}}(\boldsymbol{y})$, kde $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z}$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

4.1. Základní aplikace psaní na mokrý papír

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

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

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



Pod pojmem transformace si lze představit zejména následující.

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

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

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

Tento postup, kdy nejdříve shora uvedeným způsobem vybereme suché prvky nosiče a potom metodou psaní na mokrý papír vložíme zprávu tak, že usměrňujeme jejich zaokrouhlení, nazýváme *vkládání při kvantizaci* (perturbed quantization, PQ).

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

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

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

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

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

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

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

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

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

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

Energie bloku Energie bloku je součet čtverců všech kvantizovaných DCT koeficientů v bloku.

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

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

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

MME vkládání se od F5 vkládání liší ve čtyřech aspektech.

- Zatímco při F5 vkládání se změny v nosiči provádějí snížením absolutní hodnoty koeficientu, při MME vkládání se využívá znalost nekvantizovaného nosiče a případná změna se provede zaokrouhlením na druhé nejbližší nenulové celé číslo.

- Při MME vkládání nedochází ke smrštění, tj. $1 \mapsto 0$ nebo $-1 \mapsto 0$. Je-li potřeba změnit koeficient s hodnotou 1 nebo -1 na sudou hodnotu, použije se vždy $1 \mapsto 2$ a $-1 \mapsto -2$.

- Zatímco stegosystém F5 asociuje záporné liché AC koeficienty s hodnotou 0 a záporné sudé koeficienty s hodnotou 1, aby se vyvážilo 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 MMEd vkládání.

Poslední bod rozeberme podrobněji. Označme $\boldsymbol{x} \in \mathbb{F}_2^n$ 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 $\boldsymbol{z}$

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

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

$$\boldsymbol{y} = \boldsymbol{x} + \operatorname*{arg\,min}_{\boldsymbol{e} \in \mathcal{E}} \sum_{\substack{1 \leq i \leq n \\ e_i = 1}} \rho_i.$$

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 $\boldsymbol{z} \in \{0,1\}^m$ do nosiče $\boldsymbol{x} \in \mathbb{Z}^n$ funguje podle následujícího pravidla. Jestliže $\textsc{lsb}(x_i) = z_i$, 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	$(\ldots 00)_2$	$(\ldots 01)_2$	$(\ldots 10)_2$	$(\ldots 11)_2$
$x_i + 1$	$(\ldots 01)_2$	$(\ldots 10)_2$	$(\ldots 11)_2$	$(\ldots 00)_2$
$x_i - 1$	$(\ldots 11)_2$	$(\ldots 00)_2$	$(\ldots 01)_2$	$(\ldots 10)_2$

Nechť $\boldsymbol{H}$ 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 $\boldsymbol{x}' = (\textsc{lsb}(x_1), \ldots, \textsc{lsb}(x_n))^\mathrm{T}$ vektor nejnižších bitů nosiče $\boldsymbol{x}$ a $\boldsymbol{x}''$

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

Dvouúrovňové ± 1 vkládání má dvě fáze. V první fázi zjistíme jaké změny bude třeba provést na nejnižších bitech nosiče, čili spočteme $\boldsymbol{y}' = \mathrm{Emb}_{\boldsymbol{H}}(\boldsymbol{x}', \boldsymbol{z}')$. Na pozicích, kde se budou provádět změny jsme navíc schopni ovlivnit druhý nejnižší bit. Tyto pozice tedy označíme jako suché $\mathcal{S} = \{\, i \mid x'_i \neq y'_i \,\}$. V druhé fázi použijeme metodu psaní na mokrý papír a spočteme $\boldsymbol{y}'' = \mathrm{Emb}_{\mathcal{B}}(\boldsymbol{x}'', \mathcal{S}, \boldsymbol{z}'')$. Stegoobjekt $\boldsymbol{y}$ se vytvoří z nosiče $\boldsymbol{x}$ změnami $+1$ nebo -1 tak, aby $\boldsymbol{y}'$ byly nejnižší bity a $\boldsymbol{y}''$ druhé nejnižší bity stegoobjektu. Očekávaný 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 $r_{\mathrm{a}} = m_1/e$. Podle věty o mokrém nosiči je $m_2 \approx |\mathcal{S}| \approx r_{\mathrm{a}}$. Relativní kapacita a efektivita dvouúrovňového ± 1 vkládání je tedy

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

$$e_{\pm 1} = \frac{m_1 + m_2}{r_{\mathrm{a}}} \approx \frac{m_1 + m_1/e}{m_1/e} = e + 1. \tag{4.2}$$

Zároveň si můžeme všimnout, že

$$\frac{\alpha}{e} = \frac{r_{\mathrm{a}}}{n} = \frac{\alpha_{\pm 1}}{e_{\pm 1}}. \tag{4.3}$$

Vkládání se změnami ± 1 si spojujeme s vkládáním pomocí ternárních kódů. Jak ukazuje následující tvrzení, z hlediska efektivity dosahuje dvouúrovňové vkládání stejně dobrých výsledků jako ternární vkládání. Oproti ternárním kódů má dvouúrovňové ± 1 vkládání jednu výhodu. Při použití ternárních kódů je totiž nutné převádět mezi ternární reprezentací zpráv a obvyklou binární reprezentací, což zde odpadá.

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

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

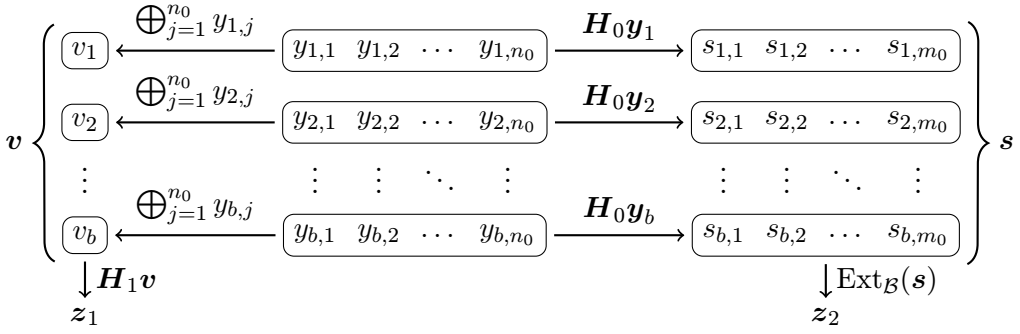
$$\alpha_{\pm 1} \approx H_2(\alpha/e) + \alpha/e = (\log_2 3) H_3(\alpha/e) = (\log_2 3) H_3(\alpha_{\pm 1}/e_{\pm 1}).$$

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

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

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

- Matice $\boldsymbol{H}_0$ vzniká z paritní matice Hammingova $[2^{m_0} - 1, 2^{m_0} - 1 - m_0]_2$ kódu přidáním nulového sloupce.
- Matice $\boldsymbol{H}_1$ je paritní matice nějakého $[b, b - m_1]_2$ kódu $\mathcal{C}$ s průměrnou vzdáleností od kódu r_{a}.



Obrázek 4.1: ZZW extrakce.

- Kódová kniha $\mathcal{B}$ umožňuje vkládat m_2 bitů do nosiče délky bm_0 bitů, z nichž $r_{\mathrm{a}}m_0$ je suchých.

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

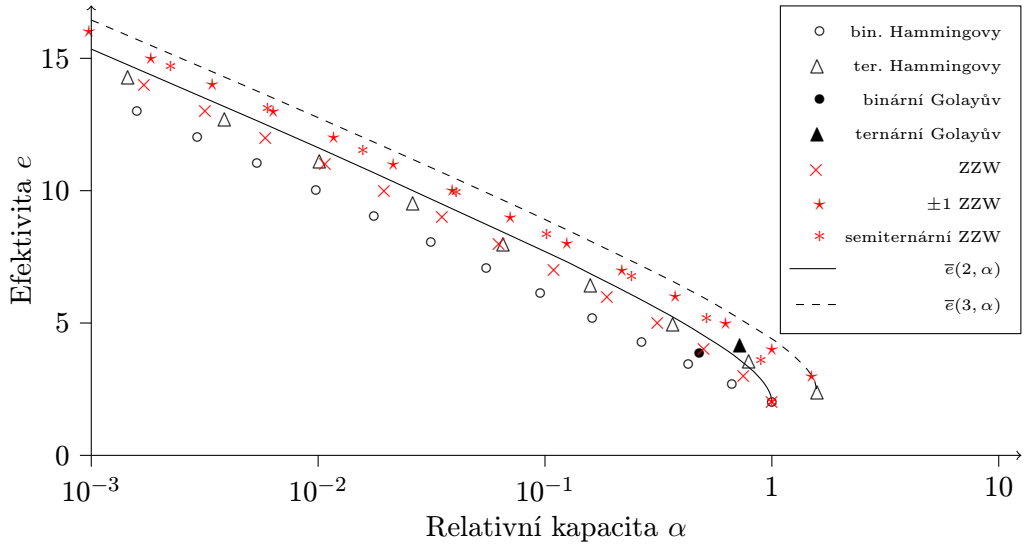
Nyní ukážeme, jak se takovýto stegoobjekt vytvoří. Vkládání dvojice zpráv $\boldsymbol{z}_1 \in \mathbb{F}_2^{m_1}$ a $\boldsymbol{z}_2 \in \mathbb{F}_2^{m_2}$ do nosiče délky $b2^{m_0}$ začneme tím, že nosič rozdělíme na b bloků velikosti $n_0 = 2^{m_0}$. Posloupnost hodnot spjatých s i-tým blokem nosiče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

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

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

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

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

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

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

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

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

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

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

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

4.3. Psaní na mokrý papír pomocí matic

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

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

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

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

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

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

```
1  S := {1,...,n} \ M
2  y_M := x_M
3  b := z − H_M x_M
4  if ( H_S | b ) má řešení then
5      za y_S zvol libovolné řešení soustavy ( H_S | b )
6      return y
7  else
8      return fail
```

Důkaz. $\boldsymbol{H}\boldsymbol{y} = \boldsymbol{H}_\mathcal{S}\,\boldsymbol{y}_\mathcal{S} + \boldsymbol{H}_\mathcal{M}\,\boldsymbol{y}_\mathcal{M} = \boldsymbol{b} + \boldsymbol{H}_\mathcal{M}\,\boldsymbol{x}_\mathcal{M} = \boldsymbol{z}$. □

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

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

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

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

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

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

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

Důkaz. Jestliže $mx \geq 1$, pak nerovnost zřejmě platí. Dále se tedy věnujme případu $mx < 1$. 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 intervalu $[0, 1/m)$ nezáporná. Funkce $f_m(x)$ je tedy neklesající na intervalu $[0, 1/m)$ a dále vidíme, že je na tomto intervalu spojitá a $f_m(0) = 0$. Odtud plyne, že $m\ln(1-x) - \ln(1-mx) \geq 0$ na $[1, 1/m)$. □

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

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

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

□

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

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

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

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

$$(\,\boldsymbol{H}_{\mathcal{S}} \mid \boldsymbol{b}\,) = \begin{array}{c} \begin{matrix} y_3 & y_4 & y_5 & y_7 \end{matrix} \\ \left(\begin{array}{cccc|c} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{array}\right) \end{array}.$$

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

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

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

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

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

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

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

$$\begin{array}{c} \begin{matrix} 1 & \ldots & k{-}1 & k & & j & \end{matrix} \\ \left(\begin{array}{ccc|cccc} 1 & * & * & * & * & * & * \\ 0 & 1 & * & * & * & * & * \\ 0 & 0 & 1 & * & * & * & * \\ \hline 0 & 0 & 0 & * & * & 0 & * \\ 0 & 0 & 0 & * & * & \mathbf{1} & * \\ 0 & 0 & 0 & * & * & 0 & * \end{array}\right) \begin{matrix} \\ \\ \\ k \\ i \\ \\ \end{matrix} \end{array} \quad \rightarrow \quad \begin{array}{c} \begin{matrix} 1 & \ldots & k{-}1 & j & & k & \end{matrix} \\ \left(\begin{array}{ccc|cccc} 1 & * & * & * & * & * & * \\ 0 & 1 & * & * & * & * & * \\ 0 & 0 & 1 & * & * & * & * \\ 0 & 0 & 0 & \mathbf{1} & * & * & * \\ \hline 0 & 0 & 0 & 0 & * & * & * \\ 0 & 0 & 0 & 0 & * & * & * \end{array}\right) \begin{matrix} \\ \\ \\ i \\ k \\ \\ \end{matrix} \end{array}$$

Algoritmus 4.10 (řešení soustavy permutacemi řádků a sloupců)**.**
vstup: soustava $(\,\boldsymbol{A} \mid \boldsymbol{b}\,)$, kde $\boldsymbol{A}$ je typu $m \times s$ nad $\mathbb{F}_q$, vektor $\boldsymbol{u} \in \mathbb{F}_q^s$
výstup: řešení soustavy, které se na alespoň $s - m$ pozicích shoduje s $\boldsymbol{u}$

$\pi := \mathrm{id} \in S_s$
for $k = 1, \ldots, m$ **do**
    v matici $\boldsymbol{A}$ najdi j-tý sloupec takový, že $w(a_{k,j}, \ldots, a_{m,j}) = 1$
    if j neexistuje **then**
        return fail
    buď $i \geq k$ takové, že $a_{i,j} \neq 0$
    swap$(\boldsymbol{A}_{i\,*}, \boldsymbol{A}_{k\,*})$
    swap(b_i, b_k)
    swap$(\boldsymbol{A}_{*\,j}, \boldsymbol{A}_{*\,k})$

```
    π := π ∘ (j, k)
for k = m, ..., 1 do
    u_π(k) := a_{k,k}^{-1} (b_k − Σ_{i=k+1}^{s} a_{k,i} u_π(i))
return u
```

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

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

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

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

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

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

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

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

Matici $\boldsymbol{H}$ typu $\mu n \times n$ nad tělesem $\mathbb{F}_q$ sestrojíme náhodně tak, aby Hammingovy váhy jejích sloupců měly robustní solitonové rozdělení s parametry $(\mu n, \delta, c)$. Při vkládání se ze sloupců této matice vybere podmatice $\boldsymbol{H}_{\mathcal{S}}$ velikosti $\mu n \times \sigma n$. Hammingovy váhy sloupců matice $\boldsymbol{H}_{\mathcal{S}}$ mají totéž rozdělení jako Hammingovy váhy sloupců matice $\boldsymbol{H}$.

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

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

Důkaz. Začneme tím, že spočítáme horní odhad na očekávanou Hammingovu váhu libovolného sloupce matice $\boldsymbol{H}$. 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

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

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

Předpokládejme, že matice $\boldsymbol{H}$ je uložena po řádcích v řídké reprezentaci. Očekávaná váha řádku matice $\boldsymbol{H}_{\mathcal{S}}$ je $O\left(\frac{s}{m}\log\frac{m}{\delta}\right)$.

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

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

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

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

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

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

5. VKLÁDÁNÍ POMOCÍ VITERBIHO ALGORITMU

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

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

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

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

5.1. Konvoluční extrakce

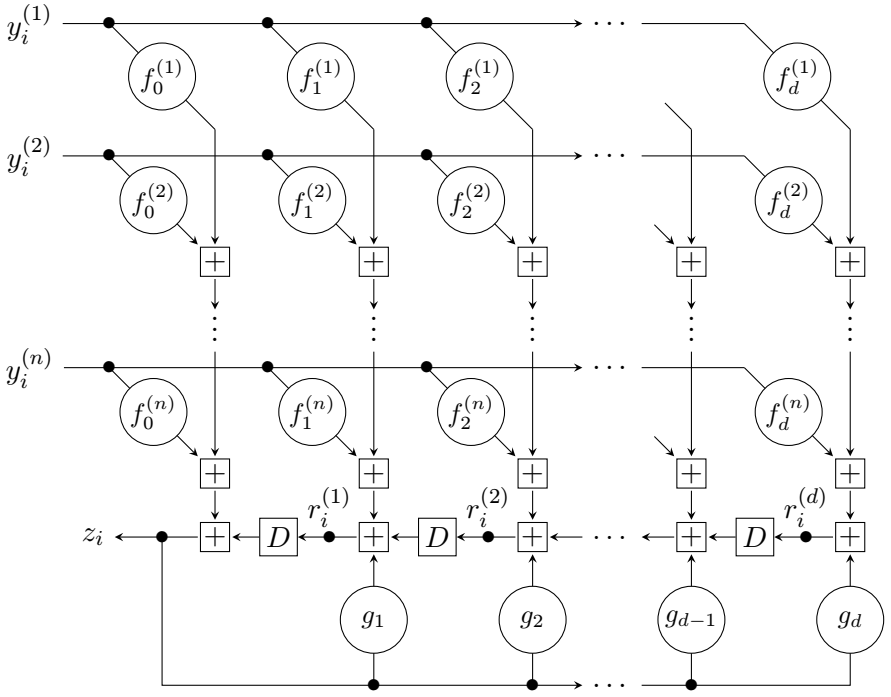
Extrakce zprávy se provádí konvolučním překladačem. Kdybychom překladač implementovali pomocí kodéru v kontrolorově normální formě, potřebovali bychom n paměťových registrů, protože v každém kroku přijímá kodér n vstupních hodnot. Překladač lze také implementovat kodérem v pozorovatelově normální formě,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

Pro referenci popíšeme extrakční algoritmus také pomocí pseudokódu.

Algoritmus 5.1 (konvoluční extrakce)**.**

vstup: stegoobjekt $\{\boldsymbol{y}_i\}_{i=0}^{\ell-1}$ z $\mathbb{F}_q^n$,
parametry $f_j^{(k)} \in \mathbb{F}_q$ a $g_j \in \mathbb{F}_q$, kde $0 \le j \le d$ a $1 \le k \le n$
výstup: zpráva $\{z_i\}_{i=0}^{\ell-1}$

1 $(s_0, \ldots, s_d) := (0, \ldots, 0)$
2 **for** $i = 0, \ldots, \ell - 1$ **do**
3 **for** $j = 1, \ldots, n$ **do**
4 $(s_0, \ldots, s_d) := (s_0, \ldots, s_d) + y_i^{(j)}(f_0^{(j)}, \ldots, f_d^{(j)})$
5 $z_i := s_0$
6 $(s_0, \ldots, s_d) := (s_0, \ldots, s_d) + s_0(g_0, \ldots, g_d)$
7 $(s_0, \ldots, s_d) := (s_1, \ldots, s_d, 0)$
8 **return** $\{z_i\}_{i=0}^{\ell-1}$



Obrázek 5.1: Konvoluční extrakce.

5.2. Souvislost s konvolučními kódy

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

$$\begin{aligned} z_i &= r_{i-1}^{(1)} + \sum_{k=1}^{n} f_0^{(k)} y_i^{(k)}, \\ r_i^{(j)} &= r_{i-1}^{(j+1)} + g_j z_i + \sum_{k=1}^{n} f_j^{(k)} y_i^{(k)}, \quad \text{pro } 1 \le j < d, \\ r_i^{(d)} &= g_d z_i + \sum_{k=1}^{n} f_d^{(k)} y_i^{(k)}. \end{aligned}$$

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

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

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

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

což po úpravě dává

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

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

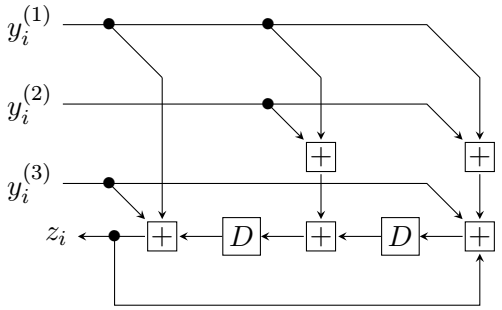
Poznámka 5.2. Kdybychom chtěli sestrojit kodér v pozorovatelově normální formě pro matici $\boldsymbol{f}$ s více řádky, stačilo by pro každý řádek sestrojit kodér popsaný na obrázku 5.1. Každý z m kodérů by měl stejný vstup a kodéry by tedy běžely paralelně vedle sebe. Jejich výstupy by se pak poskládaly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

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

5.3. Vkládání

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

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



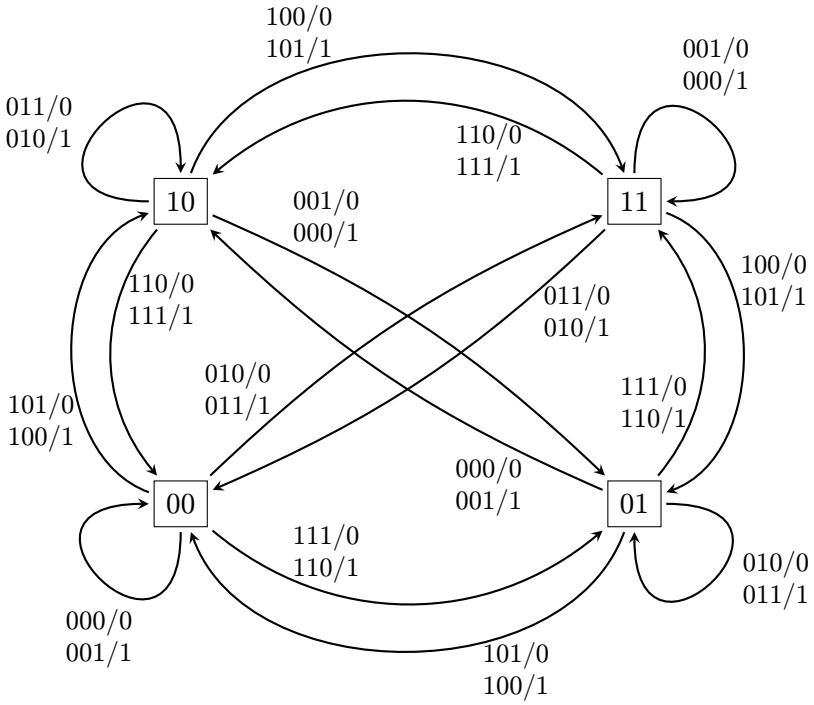
Obrázek 5.2: Konvoluční extrakce s parametry $\boldsymbol{f}(D) = \big(1 + D + D^2, D + D^2, 1 + D^2\big)$ a $g(D) = D^2$ nad $\mathbb{F}_2$.

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

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

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

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



Obrázek 5.3: Konečný překladač odpovídající kodéru na obrázku 5.2.

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

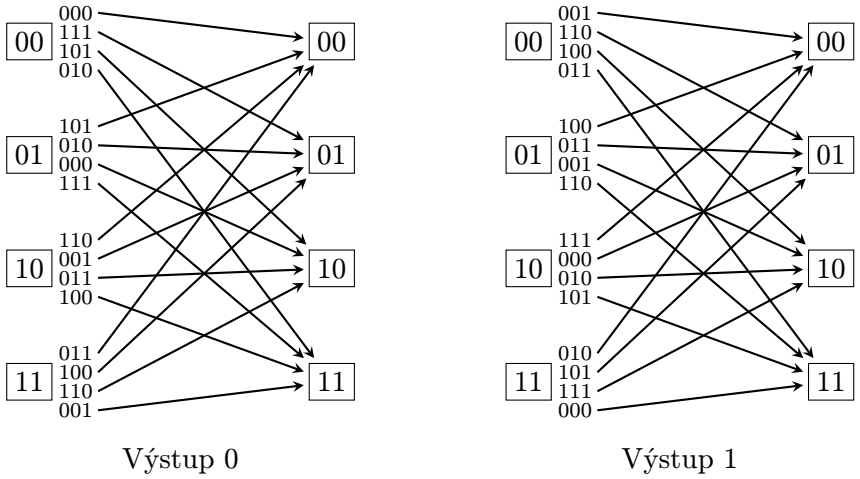
Jak už jsme zmiňovali, Viterbiho algoritmus zohledňuje váhy jednotlivých změn v nosiči. Ke každému prvku nosiče $x_i^{(j)}$ přiřazujeme distorzní funkci $\rho_i^{(j)} : \mathbb{F}_q \to \mathbb{R}$. Hodnota $\rho_i^{(j)}(a)$ udává, jak moc by přispělo nastavení $y_i^{(j)} = a$ ve stegoobjektu k celkové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

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

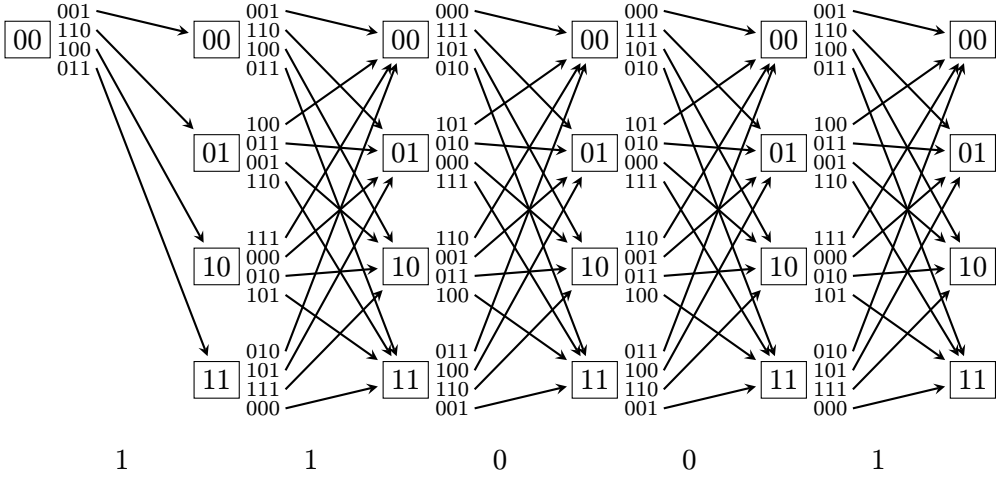
Vhodnou volbou distorzní funkce můžeme tuto metodu použít i k psaní na mokrý papír:

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

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



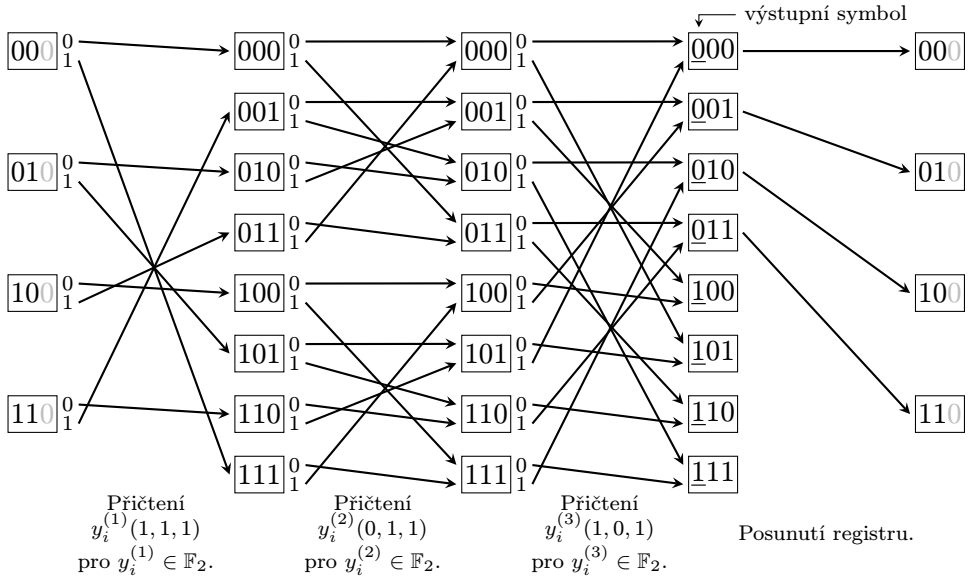
Obrázek 5.4: Trelážové moduly sestavené z konečného překladače na obrázku 5.3.



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

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

Algoritmus 5.3 (vkládání pomocí Viterbiho algoritmu)**.**
vstup: distorzní funkce prvků nosiče $\{\boldsymbol{\rho}_i\}_{i=0}^{\ell-1}$, zpráva $\{z_i\}_{i=0}^{\ell-1}$ nad $\mathbb{F}_q$, parametry extrakčního algoritmu $\boldsymbol{f}(D)$ a $g(D)$
výstup: stegoobjekt $\{\boldsymbol{y}_i\}_{i=0}^{\ell-1}$



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

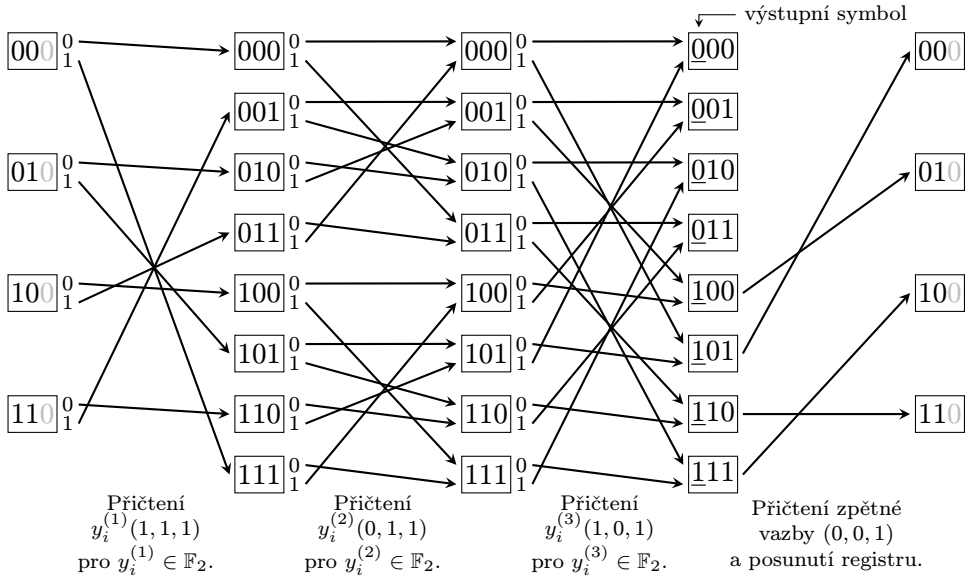
— Inicializace.
for $\boldsymbol{s} \in \mathbb{F}_q^{d+1} \setminus \{\mathbf{0}\}$ **do**
    $\mathit{weight}[\boldsymbol{s}] := \infty$
$\mathit{weight}[\mathbf{0}] := 0$

— Dopředný průchod treláží.
for $i = 0, \ldots, \ell - 1$ **do**
    for $j = 1, \ldots, n$ **do**
        for $\boldsymbol{s} \in \mathbb{F}_q^{d+1}$ **do**
            $\mathit{path}_i^{(j)}[\boldsymbol{s}] := \arg\min_{a \in \mathbb{F}_q} \mathit{weight}[\boldsymbol{s} - a(f_0^{(j)}, \ldots, f_d^{(j)})] + \rho_i^{(j)}(a)$
            $\mathit{weight}'[\boldsymbol{s}] := \min_{a \in \mathbb{F}_q} \mathit{weight}[\boldsymbol{s} - a(f_0^{(j)}, \ldots, f_d^{(j)})] + \rho_i^{(j)}(a)$
        $\mathit{weight} := \mathit{weight}'$
    for $(s_0, \ldots, s_{d-1}) \in \mathbb{F}_q^d$ **do**
        $\mathit{weight}'[(s_0, \ldots, s_{d-1}, 0)] := \mathit{weight}[(z_i, s_0, \ldots, s_{d-1}) - z_i(g_0, \ldots, g_d)]$
        for $s_d \in \mathbb{F}_q \setminus \{0\}$ **do**
            $\mathit{weight}'[(s_0, \ldots, s_d)] := \infty$
    $\mathit{weight} := \mathit{weight}'$

— Volba stavu, ve kterém končí nejlehčí cesta.
$(s_0, \ldots, s_d) := \arg\min_{\boldsymbol{s}' \in \mathbb{F}_q^{d+1}} \mathit{weight}[\boldsymbol{s}']$

— Rekonstrukce nejlehčí cesty zpětným průchodem treláže.
for $i = \ell - 1, \ldots, 0$ **do**
    $(s_0, s_1, \ldots, s_d) := (z_i, s_0, \ldots, s_{d-1}) - z_i(g_0, \ldots, g_d)$
    for $j = n, \ldots, 1$ **do**
        $y_i^{(j)} := \mathit{path}_i^{(j)}[(s_0, \ldots, s_d)]$
        $(s_0, \ldots, s_d) = (s_0, \ldots, s_d) - y_i^{(j)}(f_0^{(j)}, \ldots, f_d^{(j)})$
return $\{y_i\}_{i=0}^{\ell-1}$

Stegosystém založený na konvolučním překladači uvedeném na obrázku 5.2 dosahuje efektivity $e = 3{,}87$, přičemž $\alpha = 1/3$. Toto velmi dobře odpovídá efektivitě Hammingových 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.

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

Experimenty naznačují, že přítomnost zpětné vazby $g(D)$ nemá měřitelný vliv na efektivitu vkládání.

LITERATURA

[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, č. 3, May 1978: s. 384–386, ISSN 0018-9448, doi:10.1109/TIT.1978.1055873.
URL `http://dx.doi.org/10.1109/TIT.1978.1055873`

[2] Cohen, G.; Honkala, I.; Litsyn, S.; aj.: *Covering codes*, *North-Holland Mathematical Library*, ročník 54. Elsevier, 1997, ISBN 9780080530079.

[3] Crandall, R.: Some notes on steganography, December 1998, posted on steganography mailing list.
URL `http://os.inf.tu-dresden.de/westfeld/crandall.pdf`

[4] Filler, T.; Judas, J.; Fridrich, J.: Minimizing Additive Distortion in Steganography Using Syndrome-Trellis Codes. *Information Forensics and Security, IEEE Transactions on*, ročník 6, č. 3, Sept 2011: s. 920–935, ISSN 1556-6013, doi:10.1109/TIFS.2011.2134094.
URL `http://dx.doi.org/10.1109/TIFS.2011.2134094`

[5] Fridrich, J.: *Steganography in digital media: Principles, algorithms, and applications*. Cambridge University Press, 2010.

[6] Fridrich, J.; Goljan, M.; Soukal, D.: Perturbed quantization steganography. *Multimedia Systems*, ročník 11, č. 2, 2005: s. 98–107, ISSN 0942-4962, doi:10.1007/s00530-005-0194-3.
URL `http://dx.doi.org/10.1007/s00530-005-0194-3`

[7] Fridrich, J.; Lisoněk, P.; Soukal, D.: On Steganographic Embedding Efficiency. In *Information Hiding*, *Lecture Notes in Computer Science*, ročník 4437, editace J. L. Camenisch; C. S. Collberg; N. F. Johnson; P. Sallee, Springer Berlin Heidelberg, 2007, ISBN 978-3-540-74123-7, s. 282–296, doi:10.1007/978-3-540-74124-4˙19.
URL http://dx.doi.org/10.1007/978-3-540-74124-4_19

[8] Kim, Y.; Duric, Z.; Richards, D.: Modified Matrix Encoding Technique for Minimal Distortion Steganography. In *Information Hiding*, *Lecture Notes in Computer Science*, ročník 4437, editace J. Camenisch; C. Collberg; N. Johnson; P. Sallee, Springer Berlin Heidelberg, 2007, ISBN 978-3-540-74123-7, s. 314–327, doi:10.1007/978-3-540-74124-4˙21.
URL http://dx.doi.org/10.1007/978-3-540-74124-4_21

[9] Li, X.; Zeng, T.; Yang, B.: Improvement of the embedding efficiency of LSB matching by sum and difference covering set. In *Multimedia and Expo, 2008 IEEE International Conference on*, June 2008, s. 209–212, doi:10.1109/ICME.2008.4607408.
URL http://dx.doi.org/10.1109/ICME.2008.4607408

[10] Luby, M.: LT codes. In *Foundations of Computer Science, 2002. Proceedings. The 43rd Annual IEEE Symposium on*, 2002, ISSN 0272-5428, s. 271–280, doi:10.1109/SFCS.2002.1181950.
URL http://dx.doi.org/10.1109/SFCS.2002.1181950

[11] Zhang, W.; Zhang, X.; Wang, S.: A Double Layered "Plus-Minus One" Data Embedding Scheme. *Signal Processing Letters, IEEE*, ročník 14, č. 11, Nov 2007: s. 848–851, ISSN 1070-9908, doi:10.1109/LSP.2007.903255.
URL http://dx.doi.org/10.1109/LSP.2007.903255

[12] Zhang, W.; Zhang, X.; Wang, S.: Maximizing Steganographic Embedding Efficiency by Combining Hamming Codes and Wet Paper Codes. In *Information Hiding*, *Lecture Notes in Computer Science*, ročník 5284, editace K. Solanki; K. Sullivan; U. Madhow, Springer Berlin Heidelberg, 2008, ISBN 978-3-540-88960-1, s. 60–71, doi:10.1007/978-3-540-88961-8˙5.
URL http://dx.doi.org/10.1007/978-3-540-88961-8_5