STEGANOGRAFIE A DIGITÁLNÍ MÉDIA - ZKOUŠKOVÉ OTÁZKY

JAROSLAV KROUTIL

Tento text vznikl jako poznámky při studiu na zkoušku z předmětu *Steganografie a digitální média* (NMMB436) vyučovaném v zimním semestru školního roku 2020/2021 na MFF UK. Přednášku vedl RNDr. Andrew Kozlík Ph.D. Většina poznámek zde pochází z přednášky a webové stránky předmětu od přednášejícího, stejně jako většina obrázků a fotek. V textu se jistě vyskytuje mnoho chyb a není v něm obsažena celá odpřednášená látka.

Materiály:

- Stránka předmětu + Prezentace - Odkaz na stránku
- Oficiální seznam otázek - Odkaz na PDF
- Kozlík, A.: *Steganografie a digitální média.* - Odkaz na PDF
- Naskenované poznámky - Odkaz na stránku
- Obrázky - Odkaz na stránku
- Fridrich, J.: *Steganography in Digital media: Principles, algorithms and applications*

Obsah

Obecný přehled 2
Základní pojmy 2
Formáty obrázků (bez JPEGu) 2
Základní znalost formátu JPEG 4
Zadané otázky ke zkoušce 8
1. Popište histogramový útok na LSB embedding 8
2. Popište kvantitativní útok na Jsteg 12
3. Popište útok založený na sample pairs analysis 15
4. Popište vkládání s optimálním přiřazením parity a dokažte správnost algoritmu optimálního přiřazení parity 18
5. Popište kvantitativní útok na vkládání s optimálním přiřazením parity 25
6. Popište stegosystém OutGuess a odvoďte vzorce pro relativní kapacitu nosiče 29
7. Popište algoritmus vkládání využívaný ve stegosystému F5 a odvoďte vzorec pro relativní kapacitu nosiče bez maticového kódování 32
8. Formulujte a dokažte algoritmus maticového vkládání pomocí minimum-distance dekodéru. 35
9. Formulujte a dokažte větu o maticovém vkládání a její důsledek týkající se efektivity maticového vkládání 38
10. Odvoďte spodní mez na pokrývací poloměr v závislosti na relativní kapacitě 40
11. Odvoďte horní mez na efektivitu maticového vkládání pro kód s parametry $[n, k]_q$ a její souvislosti s perfektními kódy 43
12. Vysvětlete, co to jsou SDCS a jakým způsobem se používají ve steganografii. Odvoďte spodní mez na maximální počet změn vyvolaných SDCS vkládáním. 45
13. Formulujte a dokažte větu o mokrém nosiči. 48
14. Vysvětlete, jakým způsobem se ve steganografii využívají konvoluční kódy a Viterbiho algoritmus. 50

Obecný přehled

Základní pojmy.

Druhy strážců. Budeme posléze pracovat především s pasivním strážcem. Cílem strážce je odhalit, že dochází ke skryté komunikaci. Pokud skrytou komunikaci strážce odhalí, stegosystém se považuje za prolomený (nemusí odhalit, co je skryto, stačí jen odhalit, že skrytá komunikace probíhá).

- Pasivní strážce
  - monitoruje komunikaci a analyzuje zprávy
- Aktivní strážce
  - modifikuje komunikaci, aby zničil potenciální ukrytý obsah zpráv (např. změnou velikosti obrázku, ořezání okrajů, ztrátová komprese; změnit pořadí slov nebo nahradit synonymy)
- Zákeřný strážce
  - vkládá zprávy do utajené komunikace, vydává se za protistranu

Stegosystém.

(1) Steganografie volbou nosiče
- Máme databázi obrázků a každý obrázek má svůj tajný význam (stegoklíč = v tomto případě sada interpretačních pravidel). Např. obrázek psa = útok, obrázek kočky = ústup.

(2) Steganografie tvorbou nosiče
- např. naaranžujeme nějakou scénu, kterou vyfotografujeme (mísa s ovocem - rozmístění ovoce může skrývat zprávu)
- text: vygenerujeme zprávu, která vypadá jako spam, ale ve skutečnosti je v ní nějakým způsobem zakódována tajná zpráva

(3) Steganografie modifikací nosiče
- C množina nosičů („C“ jako cover)
- $x \in C$ nosič
- $K(x)$ množina klíčů pro daný nosič - množina nemusí být nezávislá na volbě nosiče
- $M(x)$ množina zpráv pro daný nosič (různé nosiče z C můžou mít různou kapacitu, proto je množina zpráv závislá na volbě nosiče)
- Emb: $C \times K \times M \to C$ - vkládací algoritmus (embedding)
- Ext: $C \times K \to M$ - extrakční algoritmus

$$\forall x \in C \ \forall k \in K(x) \ \forall m \in M(x) \quad \mathrm{Ext}(\mathrm{Emb}(x,k,m),k) = m$$

- Kapacita nosiče: $\log_2 |M(x)|, x \in C$ (měřená v bitech)
- Relativní kapacita nosiče: $\frac{\log_2 |M(x)|}{n}, x \in C$, kde n je počet prvků x (měříme v bpp „bits per pixel“ bitech na pixel, ale ne nutně, třeba u JPEGu to pixel není).
- Distorze (x nosič, y stegoobjekt)

$$d : C \times C \to [0, \infty)$$

$$d(\overline{x}, \overline{y}) = \sum_{i=1}^{n} (x_i - y_i)^2$$

  Říká nám, jak moc jsme změnili nosič po vložení
- Efektivní vkládání

$$e = \frac{\mathrm{E}_x \left[\log_2 |M(x)|\right]}{\mathrm{E}_{x,k,m} \left[d(x,y)\right]},$$

  kde $y = \mathrm{Emb}(x,k,m)$ (čitatel je průměrná kapacita a jmenovatel průměrná distorze). Jedná se tedy o počet vložených bitů na jednotku distorze - snažíme se maximalizovat.

Formáty obrázků (bez JPEGu).

Vektorový formát. Tento formát je nevhodný pro steganografii. Formát je založen na matematickém popisu zobrazovaných objektů a zásah do nich je komplikovaný a náchylný k chybě a odhalení.

Rastrový formát.

- např. BMP, PNG
- matice, kde každá složka obsahuje údaje o barvě příslušného pixelu
- každý pixel je tedy reprezentován trojsložkovým „vektorem“ RGB barev, tedy pokud máme hodnoty 0-255 pro každou barvu, tak máme celkově 256^3 (přes 16 milionů) barev, které můžeme použít
- bezztrátový formát
- obrázky se ale nějak komprimují a kódují, proto pokud chceme provést steganografii, tak ta se provádí přímo v obrázku a ne v zakódovaných a komprimovaných datech

Paletový formát.

- např. PNG, GIF
- matice, kde složky udávají číslo barvy v paletě
- navíc paleta, která číslům přiřazuje barvu
- paleta má obvykle 2-256 barev
- možnosti:
  (1) defaultní paleta
  (2) paleta šitá na míru obrázku
- sestrojení palety
  (1) výběr nejčastější barvy (nic moc)
  (2) Budu usilovat o to, aby i vzácné ale odlišné barvy byly v paletě (medián cut algoritmus).
  (3) snížení barevné hloubky obrázku
    (a) Každou barvu zaokrouhlím na nejbližší barvu v paletě.
    (b) Zaokrouhlím a chybu zaokrouhlení přenesu na dosud nezaokrouhlené pixely (např. Floyd-Steinberg dithering) - chybová difuze.

Obrázek 1. Porovnání použití defaultní (nahoře) a vlastní (dole) palety barev. Porovnání obyčejného nalezení nejbližší barvy z palety (vlevo) a metody chybové difuze (vpravo).

Základní znalost formátu JPEG.

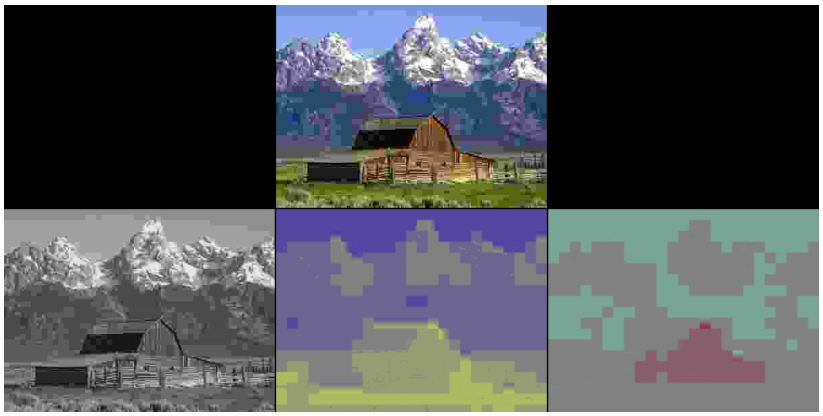
- obvykle ztrátový
- Spočívá v převodu do zcela jiné reprezentace, kde lze část dat zanedbat, aniž by se tím zásadně snížila kvalita obrázku.

JPEG kódování.

(1) Rozklad na jas Y a barvonosné složky C_b a C_r.
- Každá z těchto složek dává matici s hodnotami $\{0, 1, \ldots, 255\}$. S každou dále pracujeme samostatně.
- Každá matice tak obsahuje 1/3 informací.

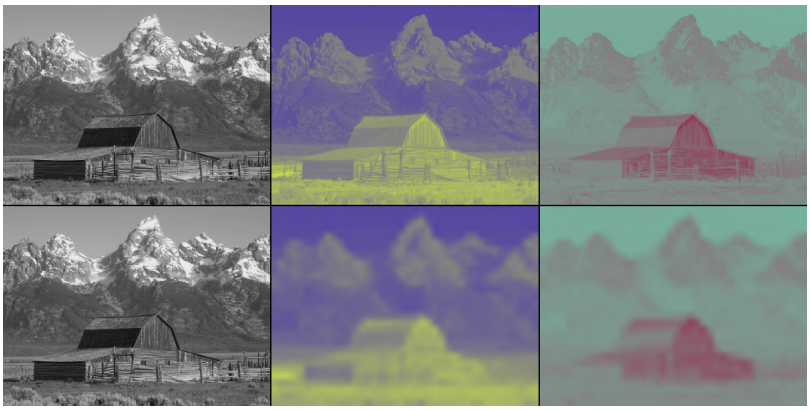
Obrázek 2. Rozklad obrázku na tři části (jas, C_b, C_r).

(2) Rozdělení na bloky a podvzorkování.
- Obraz se rozdělí na bloky 8×8 pixelů.
- Barvonosné složky se obvykle podvzorkují předtím, než jsou rozděleny na bloky.
- Pokud strany obrázku nejsou dělitelné 8 doplní se pixely, které se zase při dekódování ořežou.
- Co je podvzorkování (subsampling):
  - Neprovádí se na jas! Na jas je naše oko velmi citlivé a tedy se nekomprimuje.
  - Barvonosná část obrázku se zmenší typicky o 1/2 v každém směru.
  - Lze ovšem zmenšit jen v jednom směru nebo v žádném směru.
- Při dekódování se zase nafoukne na původní velikost.
- Pro C_b a C_r lze volit různé poměry podvzorkování.

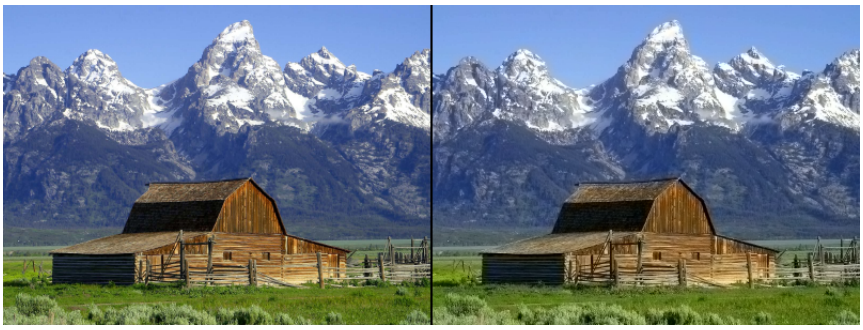


Obrázek 3. Uvažujme originální obrázek takto rozmazaný (aby šlo dobře vidět podvzorkování). Ten se rozdělí opět na jednotlivé složky a v barvonosných složkách dojde k podvzorkování. Bloky 16×16 se nahradili jednou barvou a zmenšili se na 8×8. Zde jsou už opět znovu roztažené do velikosti 16×16 po provedeném podvzorkování.

Příklad (barvonosná složka může být zkomprimována a oko prakticky nic nepozná):



Obrázek 4. V horním řádku jsou jas, C_b, C_r získané z rozkladu. Ve spodním řádku je výsledek podvzorkování C_b a C_r. To bylo provedeno zmenšením původních C_b a C_r v obou směrech 10× a poté opět zvětšeno do původní velikosti. Podvzorkované C_b a C_r tak obsahují každá pouze 1/100 původní informace.



Obrázek 5. Vlevo se nachází původní obrázek. Vpravo obrázek po podvzorkování. Ten obsahuje 34% původní informace (každá složka obsahuje 1/3 informace a dvě byly podvzorkovány, takže obsahují pouze 1/100 informace). Celkově tak máme 1/3+2/100 informace - zaokrouhleno 34%.

(3) Diskrétní kosinová transformace.

- Na začátku se od všech prvků v bloku 8×8 odečte hodnota 128 (proto, aby hodnoty byly v rozmezí -128, 127).
- Každý blok 8×8 reprezentujeme jako vektor. Máme tak 64 prvkový vektor. Takto zapsaný vektor je vlastně zapsán v kanonické bázi, která má právě 64 prvků.
- Cílem je získat novou, lepší bázi. Ortonormální bázi.
- Pro každý blok B velikosti 8×8 spočítáme blok d velikosti 8×8 jako lineární kombinaci (v bloku B tak jsou koeficienty kanonické báze a v bloku d tak budou koeficienty v nové bázi):

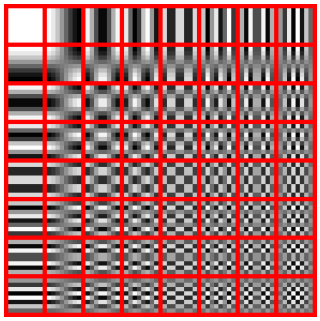
$$d_{k,l} = \sum_{i,j=0}^{7} f(i,j,k,l)B_{i,j},$$

kde

$$f(i,j,k,l) = \frac{w_k w_l}{4}\cos\left(\frac{\pi}{16}k(2i+1)\right)\cos\left(\frac{\pi}{16}l(2j+1)\right)$$

a $w_0 = \sqrt{1/2}$, $w_i = 1$ pro $i > 0$.

- Inverzní DCT je
$$B_{ij} = \sum_{k,l=0}^{7} f(i,j,k,l)d_{k,l}.$$
- Pro získání i-tého koeficientu v nové bázi stačí vynásobit blok B s i-tým vektorem z nové báze.
- Blok B 8×8 je potom kombinací bloků z báze na následujícím obrázku.



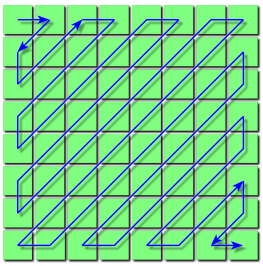
Obrázek 6. Na obrázku jsou jednotlivé prvky báze f. Koeficienty k této bázi se nacházejí v d a pomocí těchto prvků potom vznikne výsledný blok 8×8, který chceme mít na obrázku.

- K této bázi se přechází právě proto, aby se například koeficienty u detailnějších bloků (vpravo dole) mohly zanedbat. Tím pádem půjde pořád poznat, co se na obrázku nachází, ale detaily se zahodí.
- Koeficienty i, j značí souřadnice v bázovém vektoru a k, l určuje bázový vektor.
- Prvek vlevo nahoře v matici d je prakticky nejdůležitější. Určuje jas celého bloku 8×8. Tento prvek se značí DC, všechny ostatní AC (nějaká souvislost s Fourierovou řadou, střídavým a stejnosměrným proudem).
- Existují implementace DCT, které pracují s celými čísly a jsou tedy „přátelštější“ k počítačům, ale dávají malinko jiné výsledky. $\Longrightarrow$ Mohou ovlivňovat statistickou distribuci DCT koeficientů.
- DCT je vlastně změna báze v prostoru všech matic velikosti 8×8.
- Nyní je báze $\{(f(i,j,k,l)_{ij}) : \forall k,l \in \{0,1,\ldots,7\}\}$. Tato báze je ON vzhledem ke skalárnímu součinu.
- DCT koeficienty mohou teoreticky ležet v rozsahu $[-1024, 1024]$, nebývají však rovnoměrně rozděleny (koncentrovány okolo 0).

(4) Kvantizace
- Zde dochází ke ztrátě dat.
- Q je matice 8×8. Nazývá se *kvantizační matice*.
- Standard JPEG zadává sadu matic indexovaných číslem $g \in \{1,\ldots,100\}$, ale do JPEGu lze zadat vlastní kvantizační matici.
$$D_{kl} = \text{round}\left(\frac{d_{kl}}{Q_{kl}}\right),\ k,l \in \{0,\ldots,7\}$$
- Neprobíhá zde tedy žádná maticová operace ve smyslu násobení, sčítání apod.
- Čím vyšší hodnota Q_{kl}, tím více se příslušný koeficient zanedbá.
- Poté se provede rekonstrukce.
$$d'_{kl} = D_{kl}Q_{kl} \neq d_{kl}$$
nerovnost obecně neplatí, protože při výpočtu D_{kl} se zaokrouhlovalo na nejbližší celé číslo.
- Pro barvonosné složky se obvykle používá jiná kvantizační matice než pro jasovou složku.

(5) Bezztrátová komprese

- Koeficienty se převedou na posloupnost v pořadí dle následujícího obrázku. Je to z důvodu, že v levém horním rohu jsou nejdůležitější informace a v pravém dolním ty nejméně.



Obrázek 7.

- Pro každý nenulový koeficient spočítáme kolik nul mu předchází a potom ho společně s tímto údajem zakódujeme Huffmanovým kódováním. (Díky tomuto procházení máme velkou šanci na dlouhé řetězce nul).
- Tomuto se říká sekvenční kódování. (zig-zag)
- Jednotlivé bloky 8×8 potom kódujeme postupně od shora dolů.
- Takto se obrázek načítá v plné kvalitě od shora dolů.
- Alternativa je progresivní kódování. Nejdříve kódujeme koeficienty v levém horním rohu každého bloku, potom všechny koeficienty na pozici $(0, 1)$, potom všechny na pozici $(1, 0)$ (čili opět zig-zag). Toto umožňuje postupné načítání obrázku (nejdřív se načtou všechny hodnoty pomocí DC, a poté ty nejdůležitější AC a tím získáváme postupně nástin toho, co se na celém obrázku nachází, protože se postupně celý dokresluje detaily).
- Formát JPEG je mnohem obecnější, toto je jen nejtypičtější použití.

Zadané otázky ke zkoušce

1. Popište histogramový útok na LSB embedding.
Textový zdroj: naskenované poznámky - „LSB Embedding“

LSB embedding.

- Zkratka z „Least Significant Bit“, tedy vkládáme do nejnižšího bitu (např. DCT koeficientu, intenzity jasové složky nějakého koeficientu).
- Nosič je

$$x = (x_1, \ldots, x_n) \in \mathbb{Z}^n,$$

  kde n je počet prvků v obrázku (počet pixelů, počet DCT koeficientů). Jednotlivé prvky x_i pak mohou být
  - intenzita jasu i-tého pixelu
  - intenzita červené složky i-tého pixelu
  - index do palety v obrázku GIF
  - kvantizovaný DCT koeficient
- Stegoobjekt (výsledek po vkládání)

$$y = (y_1, \ldots, y_n) \in \mathbb{Z}^n$$

- Zpráva

$$z = (z_1, \ldots, z_n) \in \{0,1\}^n, \text{kde } m \leq n$$

- Klíč

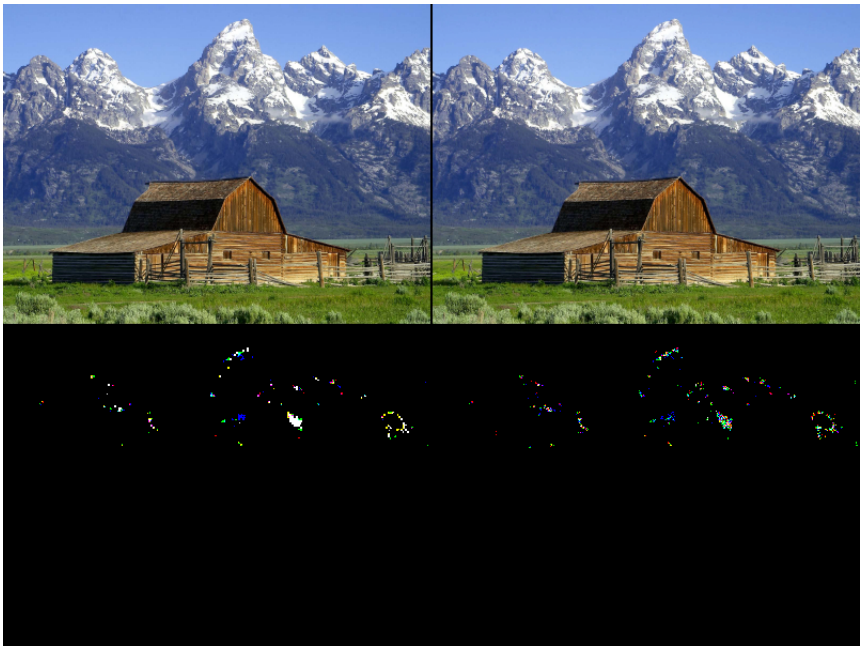
$$\pi \in S_n$$

  jedná se o permutaci na n prvcích. Říká nám do kterého pixelu bude vložen i-tý bod zprávy. Tedy pokud chci vědět, kam se zobrazí 5-tý bit zprávy, tak se podívám, kam permutace zobrazuje prvek 5.
- Vkládání. To probíhá tak, že si uděláme binární reprezentaci nosiče (např. DCT koeficientu - té dané hodnoty) a poslednímu bit dané hodnoty změníme na z_i. To lze matematicky zapsat následovně:

$$\begin{aligned} y_{\pi(i)} &= x_{\pi(i)} - (x_{\pi(i)} \bmod 2) + z_i \quad \text{pro } i = 1, \ldots, m \\ y_{\pi(i)} &= x_{\pi(i)} \quad \text{pro } i = m+1, \ldots, n \end{aligned}$$

  Tedy pokud naše $x_{\pi(i)}$ má hodnotu 3 a chceme vložit $z_i = 0$, tak $y_{\pi(i)} = 3 - (3 \bmod 2) + 0 = 3 - 1 + 0 = 2$ (binárně 0 0 1 0). Pokud chceme vložit $z_i = 1$, potom $y_{\pi(i)} = 3 - (3 \bmod 2) + 1 = 3 - 1 + 1 = 3$ (binárně 0 0 1 1).
- Platí $|x_i - y_i| \leq 1 \ \forall i$ (tedy provádíme nejmenší možnou změnu).

LSB rovina nekomprimovaných obrázků vypadá vesměs náhodně kvůli šumu při snímání obrázku. Vložení náhodné zprávy by tedy zdánlivě nemělo být detekovatelné. Ve skutečnosti existují mezi sousedními pixely vztahy, které jsou tímto vkládáním narušeny.



OBRÁZEK 8. Vlevo je originální obrázek a vpravo se nachází obrázek do nějž byla vložena zpráva délky n. Obrázky jsou pro oko nerozeznatelné. Pokud ovšem oba obrázky ztmavíme, vidíme mezi nimi rozdíly.

- Platí stěžejní vlastnost

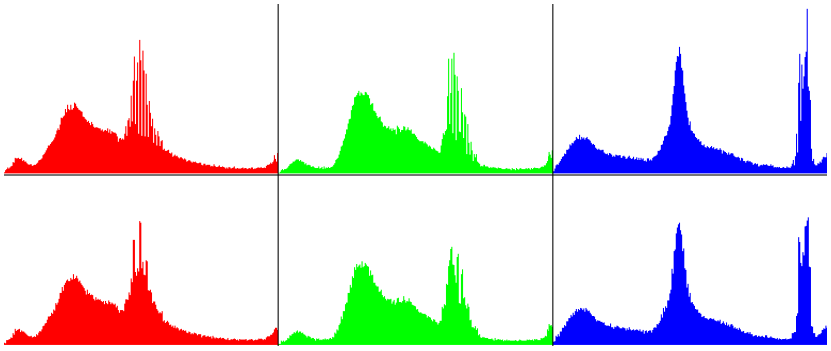
$$x_i \in \{2k, 2k+1\} \iff y_i \in \{2k, 2k+1\}.$$

Toto lze snadno nahlédnout z výpočtu vkládání.

Tato vlastnost se projeví v histogramu. Definujeme nejprve následující histogramy

$$h_0(v) = \#\{i \mid x_i = v\} \text{ histogram nosiče}$$
$$h(v) = \#\{i \mid y_i = v\} \text{ histogram stegoobjektu}$$

Tedy hodnota histogramu v bodě v je rovna počtu prvků které mají hodnotu v. Vlastnost se projeví tak, že po vložení se sloupce v histogramu seskupí do dvojic přibližně stejně vysokých sloupců - viz. následující obrázek



OBRÁZEK 9. V horní části se nacházejí histogramy pro originální obrázek. Ve spodní části pak histogramy pro obrázek, do kterého byla vložena zpráva metodou LSB.

Dále platí

$$h(2k) + h(2k+1) = h_0(2k) + h_0(2k+1)$$

díky tomu, že změny provedené vkládáním do sloupce $2k/2k+1$ se projeví jen ve sloupci $2k/2k+1$ a tedy součet histogramů těchto sloupců se nemění.

Označme nyní $\alpha = \frac{m}{n}$ (tedy pravděpodobnost, že jsme změnili pixel bitem zprávy) a předpokládejme, že vkládáme náhodnou zprávu (toto lze snadno předpokládat, protože zprávu můžeme před vložením zašifrovat (to nás nic nestojí) a výstupem šifry je náhodný řetězec), potom

$$P(y_i \neq x_i) = \alpha \cdot \frac{1}{2},$$

kde α značí pravděpodobnost, že daný pixel obsahuje bit zprávy a $1/2$ je pravděpodobnost, že vložený bit se liší od původního LSB (to díky tomu, že vkládaná zpráva je náhodná). Potom také

$$P(y_i = x_i) = 1 - \frac{\alpha}{2}$$

Díky této pravděpodobnosti máme následující vztahy

$$E\left[h(2k)\right] = \left(1 - \frac{\alpha}{2}\right) h_0(2k) + \frac{\alpha}{2} h_0(2k+1)$$

$$E\left[h(2k+1)\right] = \frac{\alpha}{2} h_0(2k) + \left(1 - \frac{\alpha}{2}\right) h_0(2k+1)$$

Histogramový útok na úplný LSB embedding. Tento útok má smysl hlavně v rastrových obrázcích. Lze pomocí něj také odhalit délku vložené zprávy. Jeho velkou chybou je, že nám neřekne s jakou pravděpodobností někoho chybně obviníme (viz konec této otázky).

Jestliže $\alpha = 1$, pak pro všechna k platí

$$E\left[h(2k)\right] = E\left[h(2k+1)\right] = \frac{h_0(2k) + h_0(2k+1)}{2} = \frac{h(2k) + h(2k+1)}{2} =: \overline{h}(2k).$$

Předpoklad $\alpha = 1$ lze taky interpretovat tak, že útočník ví v jakém pořadí se do obrázku vkládalo (v tomto případě by měřil histogram pouze v oblasti, kam se vkládalo). První rovnost platí ze vztahů pro střední hodnoty (protože $\alpha = 1$). Poslední rovnost potom ze vztahu pro hodnoty histogramů ve stegoobjektu a nosiči.

Z této rovnosti tedy plyne, že

$$h(2k) \approx \overline{h}(2k),$$

tedy sloupeček $2k$ se bude přibližně rovnat průměru sloupečku $2k$ a $2k+1$, tedy že sousední sloupečky v histogramu budou zhruba vyrovnané.

Máme-li rozhodnout, zda $\alpha = 1$ nebo $\alpha < 1$, testujeme hypotézu, že h má rozdělení $\overline{h}$:

$$H_0 : \ h \sim \overline{h} \text{ (nulová hypotéza) } \alpha = 1$$

$$H_1 : \ h \not\sim \overline{h} \text{ (alternativní hypotéza) } \alpha < 1$$

V nulové hypotéze tedy zkoumáme, zda máme stegoobjekt do kterého se vkládalo s hodnotou $\alpha = 1$. Alternativní hypotéza zahrnuje i variantu $\alpha = 0$, tedy že do objektu nebylo vůbec vkládáno.

Použijeme Pearsonův χ^2 test (chi kvadrát test). Spočítáme statistiku S:

$$S = \sum_{k=0}^{d-1} \frac{\left(h(2k) - \overline{h}(2k)\right)^2}{\overline{h}(2k)}.$$

Hodnoty přes které se sčítá záleží na tom, co řešíme. Pokud sledujeme například intenzitu červené složky v pixelu [0,255], tak $d - 1 = 128$ (zajímají nás páry). Hodnota $h(2k)$ se vztahuje k objektu, který zkoumáme.

Je zřejmé, že velká hodnota S (tedy páry sloupečků v histogramu se od sebe dost liší) indikuje $h \not\sim \overline{h}$, čili $\alpha < 1$, ale co znamená „velká hodnota“?

Hustota pravděpodobnosti rozdělení χ^2_{d-1} je

$$f_{\chi^2_{d-1}}(x) = \frac{e^{-\frac{x}{2}} x^{\frac{d-1}{2}-1}}{2^{\frac{d-1}{2}} \Gamma\left(\frac{d-1}{2}\right)}$$

čili

$$p = \int_S^\infty f_{\chi^2_{d-1}}(x)\mathrm{d}x$$

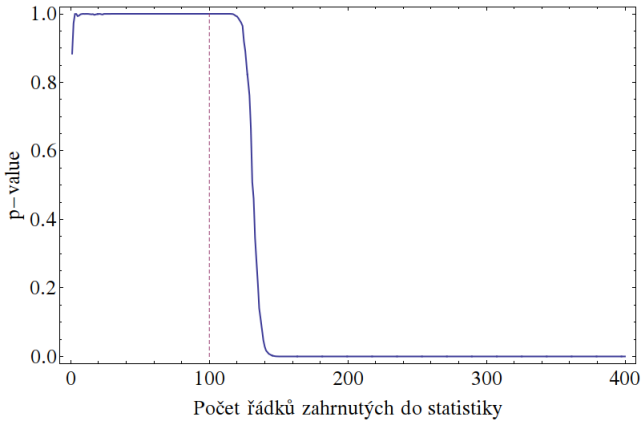
je pravděpodobnost, že S dosáhne takto vysoké (nebo) vyšší hodnoty za předpokladu $h \sim \overline{h}$.

Je-li $p \leq 0.05$ (například) pak nulovou hypotézu zamítneme a říkáme, že „zamítáme H_0 (tj. $\alpha = 1$) na hladině významnosti 0.05“. Čili

$$(S \geq 154)\ p \leq 0.05 \implies \alpha < 1$$
$$(S < 154)\ p > 0.05 \implies \alpha = 1$$

(hodnoty S jsou ilustrační)a tedy pravděpodobnost, že chybně zamítneme hypotézu $\alpha = 1$ je 5%.

Tímto útokem lze odhadnout délku zprávy. Předpokládáme, že útočník ví, v jakém pořadí byla do obrázku zpráva vkládána (zná klíč - permutaci), ale neví délku zprávy a chce ji odhadnout. Předpokládejme, že se do obrázku vkládalo postupně po řádcích (tedy nejprve do pixelů v prvním řádků, pak v druhém atd.). Následující graf ukazuje, jak lze odhad provést. Budeme si postupně brát řádky a počítat pravděpodobnost. Vidíme, že se pravděpodobnost drží okolo 1 a pak prudce klesne. Zpráva má délku 100 (to útočník neví), ale vidí, že okolo 140-tého řádku pravděpodobnost prudce klesla. Tudíž vidí, že délka zprávy je menší, než 140. I když je délka 100, tak se pravděpodobnost ještě nějakou dobu drží okolo 1.



Obrázek 10.

Dále z grafu vidíme, že pokud je zpráva velmi krátká (jen na prvních řádcích - tedy např. 2/400), tak chi kvadrát test není dost citlivý na to, aby dokázal tak krátkou zprávu zachytit (stále předpokládáme, že útočník zná klíč - tedy onu permutaci podle které se do obrázku vkládá).

2. Popište kvantitativní útok na Jsteg.
Textový zdroj: naskenované poznámky - „LSB Embedding“

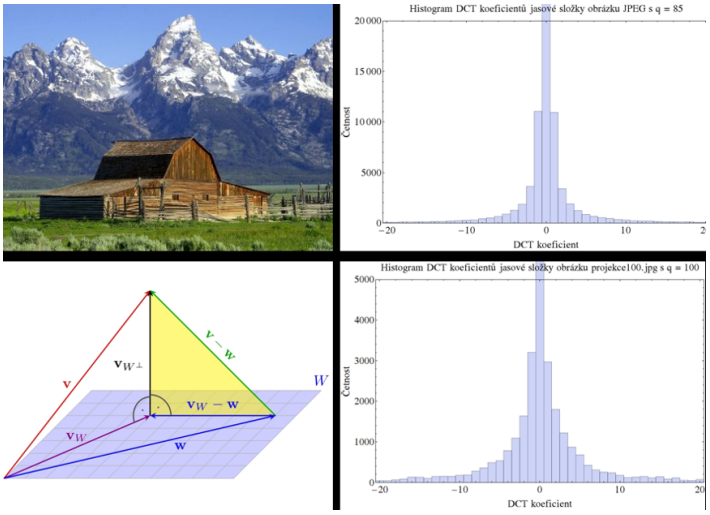
Kvantitativní znamená, že určíme α. Jsteg vkládá zprávu do LSB DCT koeficientů obrázku JPEG, s výjimkou koeficientů s hodnotami 0 a 1. Vkládání do 0 se totiž projeví vizuálně. Pokud bychom nevkládali do 0, ale vkládali do 1 (a ostatních koeficientů), potom by mohlo dojít ke změně některých koeficientů z 1 na 0 ale příjemce by nevěděl, které 0 koeficienty byly změněny z 1 a které byly i původně 0. Pořád totiž platí vlastnost

$$x_i \in \{2k, 2k+1\} \iff y_i \in \{2k, 2k+1\}$$

a tedy pokud vynecháme vkládání do koeficientů s hodnotami 0 a 1, tak se obrázek skoro vůbec nezmění a příjemce bude schopen zprávu získat.

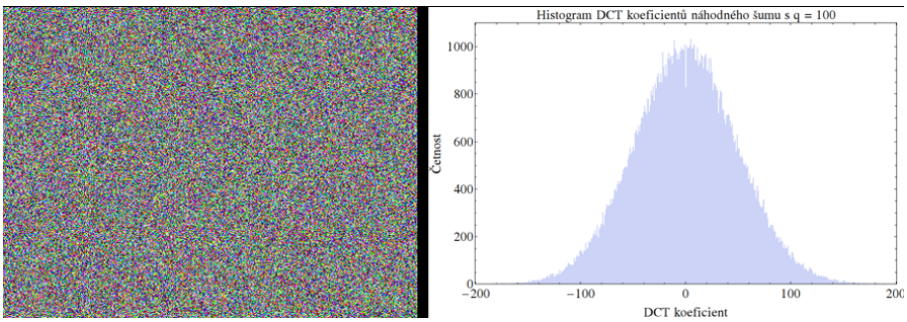
Obrázek 11. Vlevo se nachází obrázek do nějž byla vložena zpráva do každého DCT koeficientu. Vpravo potom obrázek po vložení zprávy Jstegem (tedy vynecháním koeficientů s hodnotami 0 a 1).

Nyní se podívejme na histogramy DCT koeficientů u JPEG obrázků.

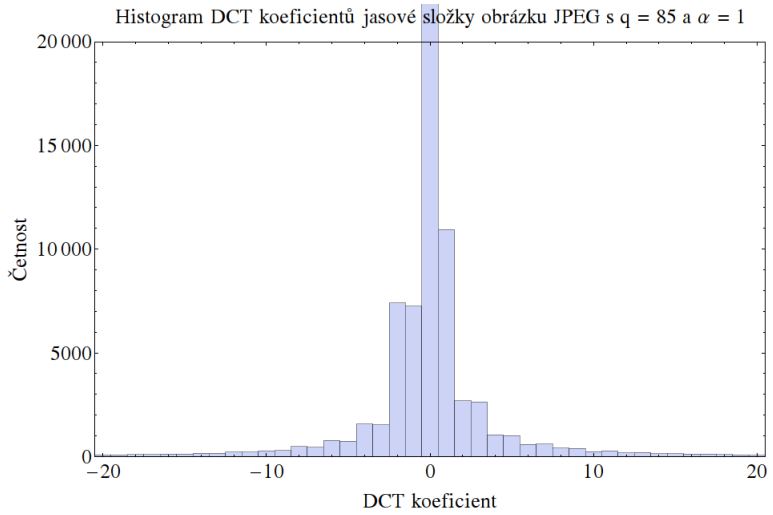


Obrázek 12. Vlevo vidíme dva naprosto odlišné obrázky, jeden je fotografie a druhý je matematický nákres. Přesto tyto obrázky mají velmi podobné histogramy. Histogramy se s klesající kvalitou mění tak, že všechny sloupce kromě 0 se zmenšují. Zároveň je patrná jistá symetričnost pro DCT koeficienty i a $-i$.

Pro zajímavost, takto vypadá histogram pro náhodný šum ve 100% kvalitě. Histogram má normální rozdělení. I zde ale platí, že s klesající kvalitou se ostatní sloupce, kromě nulového, zmenšují.



Obrázek 13.



Obrázek 14. Histogram po vložení zprávy do obrázku s $\alpha = 1$.

Víme tedy, že DCT koeficienty v nezměněném JPEG obrázku mají symetrický histogram a tedy

$$h_0(i) \approx h_0(-i) \quad i = 1, 2, \ldots$$

a proto platí

$$\sum_{k>0} h_0(2k) - \sum_{k<0} h_0(2k) - \sum_{k\geq 0} h_0(2k+1) + \sum_{k<0} h_0(2k+1) \approx 0$$

Tento vztah neříká nic jiného, než že když od sebe odečteme sloupec i a $-i$ tak bychom měli obdržet přibližně nulu. Drobné upozornění - ve třetí sumě je neostrá nerovnost, protože i sloupec pro koeficient 1 je potřeba zahrnout.

Nyní se opět podívejme na soustavu rovnic (z minulé otázky)

$$E\left[h(2k)\right] = \left(1 - \frac{\alpha}{2}\right) h_0(2k) + \frac{\alpha}{2} h_0(2k+1)$$
$$E\left[h(2k+1)\right] = \frac{\alpha}{2} h_0(2k) + \left(1 - \frac{\alpha}{2}\right) h_0(2k+1)$$

Z těchto rovnic můžeme vyjádřit $h_0(2k)$ a $h_0(2k+1)$.

$$h_0(2k) = \frac{1-\frac{\alpha}{2}}{1-\alpha}E[h(2k)] - \frac{\frac{\alpha}{2}}{1-\alpha}E[h(2k+1)]$$
$$h_0(2k+1) = -\frac{\frac{\alpha}{2}}{1-\alpha}E[h(2k)] + \frac{1-\frac{\alpha}{2}}{1-\alpha}E[h(2k+1)]$$

Můžeme si označit $a = \frac{1-\alpha/2}{1-\alpha}$ a $b = \frac{\alpha/2}{1-\alpha}$ a tedy

$$h_0(2k) = a \cdot E[h(2k)] - b \cdot E[h(2k+1)]$$
$$h_0(2k+1) = -b \cdot E[h(2k)] + a \cdot E[h(2k+1)]$$

Provedeme aproximaci $h(i) \approx E[h(i)]$ a tedy

$$h_0(2k) = a \cdot h(2k) - b \cdot h(2k+1)$$
$$h_0(2k+1) = -b \cdot h(2k) + a \cdot h(2k+1)$$

Toto nyní můžeme dosadit do vztahu, kde jsem od sebe odečítali sloupce z histogramu

$$\sum_{k>0} a{\cdot}h(2k){-}b{\cdot}h(2k{+}1){-}\sum_{k<0} a{\cdot}h(2k){-}b{\cdot}h(2k{+}1){-}\sum_{k>0} {-}b{\cdot}h(2k){+}a{\cdot}h(2k{+}1){+}\sum_{k<0} {-}b{\cdot}h(2k){+}a{\cdot}h(2k{+}1) \approx h_0(1) = h(1)$$

Ve třetí sumě již není neostrá nerovnost, protože jsme (pro $k = 0$) $h_0(1)$ přesunuli na pravou stranu. Rovnost $h_0(1) = h(1)$ platí, protože jsme do koeficientů s hodnotou 1 nic nevkládali. Z toho potom máme

$$(a+b)\sum_{k>0}(h(2k) - h(2k+1)) + (a+b)\sum_{k<0}(h(2k+1) - h(2k)) \approx h(1),$$

kde $a + b = \frac{1}{1-\alpha}$.

Z toho můžeme vyjádřit α a tedy

$$\alpha \approx 1 - \frac{\sum_{k>0}(h(2k) - h(2k+1)) + \sum_{k<0}(h(2k+1) - h(2k))}{h(1)}.$$

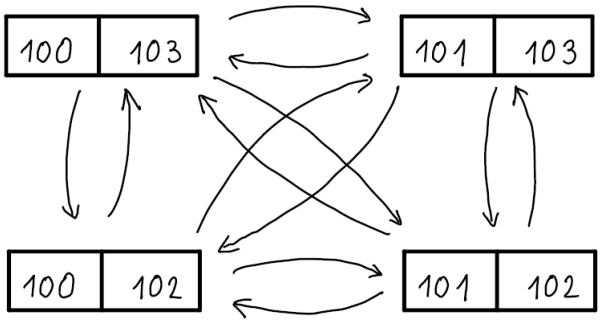
Hodnota α obvykle vychází s přesností ± 0.05. Z toho plyne, že vkládat do 5% obrázku není touto metodou detekovatelné.

3. Popište útok založený na sample pairs analysis.
Textový zdroj: naskenované poznámky - „LSB Embedding“ (POZOR! na 4. straně od „Obecně:“ není potřeba umět včetně celé 5. strany; 6. stranu už ano)

Sample pairs analysis - jedná se o analýzu dvojic vzorků. Je určena pro rastrové obrázky. Nejprve si zopakujme princip histogramového útoku (chi kvadrát test). Ten funguje pro $\alpha = 1$ vs $\alpha \neq 1$. Pro obecné α bychom potřebovali vědět více o tvaru histogramu nosiče: Problém je v tom, že histogramy nosičů jsou příliš různorodé na to, abychom mohli vypreparovat nějakou obecnou vlastnost.

Základem analýzy je podívat se na páry sousedních pixelů. V přirozeném obrázku očekáváme, že sousední pixely budou mít blízké hodnoty. Například:

Co se stane při LSB: (výskyt se díky použití LSB vyváží)



Jde v podstatě o to, že histogram párů sousedních pixelů má předvídatelný tvar. Přesněji jde o to, že v nosiči by výskyty těchto čtyř párů měly být vyvážené.

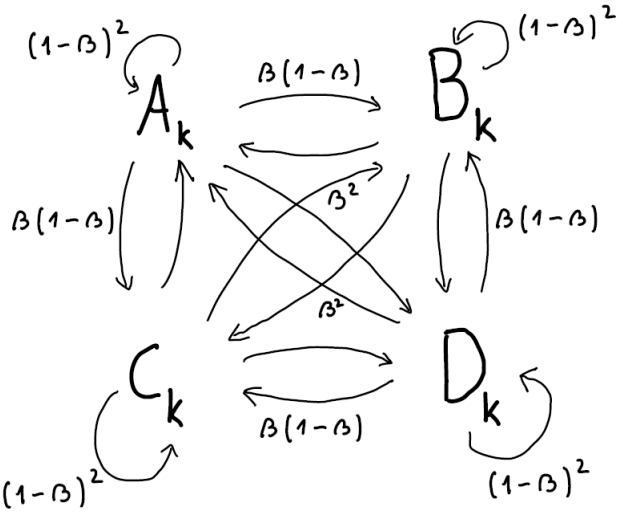
Modifikovaná Sample Pairs analysis. Označme

$$\begin{aligned} A_k &= \{(2i, 2j) \in P \mid i - j = k\} \\ B_k &= \{(2i, 2j+1) \in P \mid i - j = k\} \\ C_k &= \{(2i+1, 2j) \in P \mid i - j = k\} \\ D_k &= \{(2i+1, 2j+1) \in P \mid i - j = k\}, \end{aligned}$$

kde P je multimnožina (jeden prvek se zde může nacházet vícekrát) všech párů sousedících pixelů. Páry můžeme najít procházením pořádcích a zároveň pak i po sloupečcích pro všechny barevné složky. Dvojice je pak tvořena hodnotami dané složky v daných pixelech. Prvky A_k, B_k, C_k, D_k jsou taktéž multimnožiny.

Podíváme-li se na předchozí dvojici $(100, 102)$, tak tato dvojice by spadla do A_k pro $k = 1$.

Zavedeme ještě pravděpodobnost toho, že dojde ke změně při vložení. Budeme ji značit β a je rovna $\beta = \frac{\alpha}{2}$. Tento vztah vychází z toho, že pravděpodobnost, že se zpráva zapíše do daného bitu je $\alpha = \frac{m}{n}$ a pravděpodobnost, že bude bit změněn je $\frac{1}{2}$, protože předpokládáme, že zpráva je náhodná. Potom $1 - \beta$ je pravděpodobnost že bit nepotřebujeme měnit. Díky tomu máme následující schéma



Obrázek 15. Například z A_k do A_k je pravděpodobnost $(1-\beta)^2$, protože ani v jedné souřadnici nedošlo ke změně a pravděpodobnost že nedojde ke změně v jedné souřadnici je $(1-\beta)$. Analogicky u ostatních šipek.

Označme nyní a_k, b_k, c_k, d_k velikosti multimnožin před vkládáním (tyto hodnoty neznáme) a a'_k, b'_k, c'_k, d'_k jejich velikosti po vkládání (tyhle hodnoty máme, protože se jedná o hodnoty se stegoobjektu). Potom

$$\begin{pmatrix} a'_k \\ b'_k \\ c'_k \\ d'_k \end{pmatrix} \approx \begin{pmatrix} (1-\beta)^2 & \beta(1-\beta) & \beta(1-\beta) & \beta^2 \\ \beta(1-\beta) & (1-\beta)^2 & \beta^2 & \beta(1-\beta) \\ \beta(1-\beta) & \beta^2 & (1-\beta)^2 & \beta(1-\beta) \\ \beta^2 & \beta(1-\beta) & \beta(1-\beta) & (1-\beta)^2 \end{pmatrix} \cdot \begin{pmatrix} a_k \\ b_k \\ c_k \\ d_k \end{pmatrix}.$$

Z toho můžeme vyjádřit prvky a_k, b_k, c_k, d_k

$$\begin{pmatrix} a_k \\ b_k \\ c_k \\ d_k \end{pmatrix} \approx \frac{1}{1-2\beta} \begin{pmatrix} (1-\beta)^2 & -\beta(1-\beta) & -\beta(1-\beta) & \beta^2 \\ -\beta(1-\beta) & (1-\beta)^2 & \beta^2 & -\beta(1-\beta) \\ -\beta(1-\beta) & \beta^2 & (1-\beta)^2 & -\beta(1-\beta) \\ \beta^2 & -\beta(1-\beta) & -\beta(1-\beta) & (1-\beta)^2 \end{pmatrix} \cdot \begin{pmatrix} a'_k \\ b'_k \\ c'_k \\ d'_k \end{pmatrix}.$$

Nyní nás budou zajímat rovnice pro b_k a c_k. Proč? Mezi B_k a C_{k-1} máme jistý vztah. V původním nezměněném obrázku platí $B_k = \{(2i, 2j+1) \in P \mid i - j = k\}$, kde pro vzdálenost platí $2i - (2j+1) = 2(i-j) - 1 = 2k - 1$ a $C_{k-1} = \{(2i+1, 2j) \in P \mid i - j = k-1\}$, kde vzdálenost platí $2i + 1 - 2j = 2(i-j) + 1 = 2(k-1) + 1 = 2k - 1$. Tedy vzdálenosti mezi prvky v dvojicích se rovnají.

Očekáváme, že v nosiči platí $|B_k| \approx |C_{k-1}|$, protože páry v obou třídách mají vzdálenost $2k-1$ a liší se pouze paritou. Parita by v nosiči neměla mít vliv na četnost.

Podíváme-li se na 2. a 3. řádek soustavy, dostáváme pro každé k kvadratickou rovnici

$$\begin{aligned} b_k &\approx \frac{1}{1-2\beta}\left(a'_k(-\beta(1-\beta)) + b'_k(1-\beta)^2 + c'_k\beta^2 + d'_k(-\beta(1-\beta))\right) \\ &\approx \frac{1}{1-2\beta}\left(\beta^2(a'_k + b'_k + c'_k + d'_k) - \beta(a'_k + 2b'_k + d'_k) + b'_k\right) \\ c_{k-1} &\approx \frac{1}{1-2\beta}\left(a'_{k-1}(-\beta(1-\beta)) + b'_{k-1}(1-\beta)^2 + c'_{k-1}\beta^2 + d'_{k-1}(-\beta(1-\beta))\right) \\ &\approx \frac{1}{1-2\beta}\left(\beta^2(a'_{k-1} + b'_{k-1} + c'_{k-1} + d'_{k-1}) - \beta(a'_{k-1} + 2c'_{k-1} + d'_{k-1}) + c'_{k-1}\right) \end{aligned}$$

Prvek $\frac{1}{1-2\beta}$ se odstaví a zůstane kvadratická rovnice. Protože očekáváme $|B_k| \approx |C_{k-1}|$, tak pro naše kvadratické rovnice platí $b_k \approx c_{k-1}$.

Vlastnost $|A_k| \approx |D_k|$ se využít nedá, pouze to implikuje, že $|A'_k| \approx |D'_k|$ $\forall \beta$ (tedy triviální informace).

Jak naložit s touto sadou kvadratických rovnic? (sadou, protože máme rovnice pro každé k)

(1) Pro např. $k = -2, -1, 0, 1, 2$ najdeme kořeny kvadratické rovnice a z nich vybereme minimální.
   - Některé kořeny, které nám vyjdou, ani nemusí splňovat požadavky na β ($\beta \in [0, \frac{1}{2}]$).
   - Minimální řešení vybíráme proto, protože se může vyskytnout nějaké řešení (outlier), které je pouze důsledkem šumu.
   - Tím že volíme minimální řešení, tak vlastně předpokládáme, že námi zkoumaný objekt je bez zásahu (nebylo do něj nic vloženo).

(2) Kvadratické rovnice sečteme pro všechna k sudé, nebo pro k liché a dostaneme tak jednu rovnici a tu vyřešíme.
   - Kdybychom sečetli přes všechna k, tak se nám průběžně „požírají“ některé členy a ve výsledku dostaneme lineární rovnici jejímž řešením je $\beta = 0.5$.

(3) Nebudeme vůbec řešit kvadratické rovnice tj. nebudeme předpokládat $b_k = c_{k-1}$, ale zvolíme β tak, aby b_k bylo co nejblíže c_{k-1} pro všechny k (metodou nejmenších čtverců)

$$\hat{\beta} = \operatorname{argmin}_\beta \sum_k (b_k - c_{k-1})^2,$$

kde za b_k a c_{k-1} dosadíme výrazy z kvadratických rovnic pro tyto prvky (viz. výše).

Pokus s $\beta = 0.2$: Průměr odchylky $|\beta - \hat{\beta}|$ (tedy skutečné hodnoty β od spočítané hodnoty $\hat{\beta}$) pro různé metody.

- SPA (klasická verze Sample Pairs Analysis): 0.0077
- (1): 0.0466
- (2) $k = -50, \ldots, 50$ sudé: 0.0101
- (3) $k = -50, \ldots, 50$: 0.0056

Tedy nejlépe vychází výpočet třetím způsobem, ovšem tento výpočet je nejnáročnější.

4. Popište vkládání s optimálním přiřazením parity a dokažte správnost algoritmu optimálního přiřazení parity.
Textový zdroj: naskenované poznámky - „LSB Embedding“, prezentace - „Steganografie v paletových obrázcích“

V této otázce se budeme zabývat paletovými obrázky. Nejprve si představíme nějaké základní vlastnosti a možné steganografické implementace v paletových obrázcích, abychom měli přehled o tom, co se to vlastně děje.

Reprezentace barev. Barvy se v paletovém obrázku reprezentují jako uspořádané trojice $(r, g, b) \in \{0, \ldots, 255\}^3$. Množinu všech barev lze uspořádat lexikograficky jako množinu trojic. Podobnost barev můžeme měřit pomocí Euklidovské normy

$$||(r_1, g_1, b_1) - (r_2, g_2, b_2)|| = \sqrt{(r_1 - r_2)^2 + (g_1 - g_2)^2 + (b_1 - b_2)^2}.$$

Existují i jiné reprezentace barev, které mohou být vhodnější pro měření podobnosti, např. CIELAB. CIELAB rozkládá barvy na jasovou složku a dvě barvonosné složky. Transformace mezi RGB a CIELAB je nelineární.

Paletové obrázky. Paletový obrázek sestává z palety barev a matice indexů. Paleta barev je množina $C = \{c_i \mid i \in I\}$ indexovaná čísly $I = \{0, 1, \ldots, |C| - 1\}$. Barva pixelu je určena indexem na odpovídající pozici v matici indexů. Každý paletový obrázek má mnoho různých reprezentací určených uspořádáním palety.

Steganografie v paletových obrázcích.

(1) Volbou uspořádáním palety
(2) Vkládáním do matice indexů
    (a) Paletu setřídíme podle jasu. Potom LSB embedding.
    (b) Paletu setřídíme lépe. Potom LSB embedding.
    (c) Vkládání s optimálním přiřazením parity.

Steganografie volbou uspořádání. Množinu všech permutací palety S_C můžeme uspořádat lexikograficky. Každé permutaci přiřadíme pořadové číslo od 0 do $|C|! - 1$. Každému pořadovému číslu přiřadíme nějakou zprávu. Velkou výhodou této metody je, že nezanechává žádnou vizuální stopu v obrázku (jen se zpermutuje paleta a tím pádem se změní i index, ale barvy na které indexy odkazují se nemění). Nevýhodami jsou:

- Omezená kapacita
- Chaoticky uspořádaná paleta budí podezření (většinou totiž bývá setříděna podle jasu, odstínu, anebo podle četnosti barvy).
- Otevření obrázku a opětovné uložení má zpravidla za následek zničení zprávy (uložením se přerovná paleta do standardní podoby).
- Lze aplikovat forma útoku, kdy se útočník ani nesnaží zprávu najít, ale rovnou vše modifikuje (přerovnává).

Vkládání do matice indexů (2.a). Algoritmem EzStego. Paletu setřídíme podle jasu a matici indexů přečíslujeme. Bity zprávy vkládáme do nejnižších bitů jednotlivých indexů v matici. Změna nejnižšího bitu v indexu příliš neovlivní jas (paleta je setříděna podle jasu), ale může vést k zásadní změně odstínu pixelu.

OBRÁZEK 16. Vlevo nahoře se nachází originální obrázek. Vpravo nahoře obrázek po vložení zprávy pomocí 2.a. Vlevo dole se nachází pouze jasová složka z originálního obrázku. Vpravo dole se nachází jasová složka obrázku s vloženou zprávou.

Vkládání do matice indexů (2.b). Jedná se o vkládání s lépe setříděnou paletou. Postupujeme obdobně jako v 2.a, ale paletu uspořádáme, aby každé dvě po sobě následující barvy v paletě byly co možná nejpodobnější. To můžeme udělat například tak, že budeme minimalizovat součet

$$\sum_{i=0}^{|C|-2} ||c_i - c_{i+1}||,$$

což je minimalizování vzdálenosti dvou barev v krychli s RGB stěnami (c_i má tři složky, takže minimalizujeme Euklidovskou normu zmíněnou na začátku této otázky). Problém hledání takovéhoto uspořádání je shodný s problémem obchodního cestujícího (NP-úplný problém).

Odesílání potom probíhá následovně:

- Nejprve si odesílatel a příjemce musí domluvit na nějakém deterministickém algoritmu, který přeuspořádá paletu na ono požadované přeuspořádání palety (vzdálenosti dvou barev totiž mohou být stejné, takže potřebují, aby oba měli stejnou paletu a nemohou se tak oba nezávisle pokusit vyřešit onu zmíněnou minimalizaci).
- Odesílatel vezme nosič který přečísluje podle přeuspořádané palety.
- Odesílatel vloží do matice indexů zprávu tím, že upraví vždy poslední bit indexu. Tím dojde opět ke změně barev, ovšem změna nebude tolik drastická jako v 2.a, protože jsou barvy seřazeny tak, aby ležely co nejblíže sebe a tedy odstín barvy by se změnou posledního bitu v indexu neměl příliš změnit.
- Nyní odesílatel přeuspořádá paletu do nějaké obvyklé podoby (třeba ji setřídí podle jasu) a upraví taky příslušnou matici indexů.
- Odesílatel odešle obrázek s vloženou zprávou.

- Příjemce paletu přeuspořádá pomocí algoritmu na kterém se s odesílatelem domluvili a upraví matici indexů (tyto indexy jsou nyní shodné s těmi na které je upravil odesílatel po vložení zprávy).
- Příjemce se podívá na poslední bit každého indexu a takto získá zprávu.

Vkládání s optimálním přiřazením parity (2.c). Idea: Každé barvě v paletě přiřadíme hodnotu 0 nebo 1 (paritu) tak, aby nejpodobnější barvy v paletě měla opačnou hodnotu (to proto, aby se změna v indexu co nejméně projevila). Potřebujeme-li, aby barva některého pixelu měla opačnou paritu, stačí ji změnit na nejpodobnější barvu (ta má právě opačnou paritu). Tímto minimalizujeme velikost změn vyvolaných vkládáním.

Pro toto vkládání potřebujeme následující:

- Zobrazení, které každé barvě přiřazuje paritu, budeme značit p.
- Zobrazení, které každé barvě přiřazuje nejpodobnější jinou barvu v paletě, budeme značit f (flip).

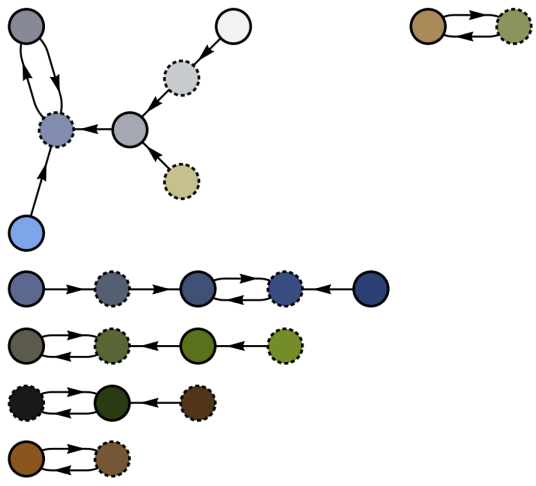
Všimněme si, že pokud bude zobrazení p splňovat, že nejpodobnější barvy budou mít opačnou paritu, tak přechodem od barvy k nejpodobnější barvě pomocí f dojde zároveň ke změně parity.

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 *optimálním přiřazením parity* pro paletu C.

Cílem je, aby bylo p a f jednoznačně určitelné bez předchozí dohody. Tedy aby si odesílatel a příjemce byli schopni p a f spočítat pouze z palety barev (libovolně uspořádané). Podívejme se na příklad takového p a f v následujícím grafu barev z palety:



Obrázek 17. Zobrazení p je v grafu znázorněno celistvým/přerušovaným kruhem okolo barvy. Zobrazení f je znázorněno šipkami mezi barvami.

Jak jednoznačně určit p a f si ukážeme později. Nyní se podívejme na algoritmy vkládání a extrakce.
Algoritmus vkládání s optimálním přiřazením parity (předpokládáme znalost p a f):

```
VSTUP:
        nosič x = (x_1, ..., x_n) ∈ I^n        % x_i značí index do palety
        zpráva z = (z_1, ..., z_m) ∈ {0,1}^m
        klíč π ∈ S_n
        optimální přiřazení parity (p, f)
VYSTUP:
        stegoobjekt y = (y_1, ..., y_n) ∈ I^n
ALGORITMUS:
for 1, ..., n do
    if i > m or p(x_π(i)) = z_i then
        y_π(i) := x_π(i)
    else
        y_π(i) := f(x_π(i))
return y
```

Vkládání zprávy probíhá změnou parity. Podmínka $i > m$ nám říká, že jsem již mimo zprávu a podmínka $p(x_{\pi(i)}) = z_i$ nám říká, že parita barvy, kam chceme vkládat zprávu, má stejnou hodnotu jako bit zprávy z_i, který zde chceme vložit. V obou případech tedy není potřeba nic měnit a do stegoobjektu tak vložíme stejný index jako je v nosiči.

Pokud tedy $i < m$ a parita má jinou hodnotu než bit z_i, který chceme vložit, tak musíme změnit paritu, což provedeme pomocí zobrazení f. To vede na nejbližší barvu (takže se změní odstín jak nejméně to jde). Změníme tedy index. Ten nyní bude ukazovat na nejbližší barvu, která zároveň z definice splňuje, že má opačnou paritu (tedy hodnotu vkládaného bitu z_i). Celkově tak do $y_{\pi(i)}$ uložíme index odkazující na nejbližší barvu k barvě na kterou odkazuje index $x_{\pi(i)}$.

Všimněme si také, že algoritmus nezávisí na uspořádání palety, protože parita je těsně svázaná s konkrétní barvou a nezáleží na jejím umístění v paletě.
Algoritmus extrakce s optimálním přiřazením parity (předpokládáme znalost p a f):

```
VSTUP:
        stegoobjekt y = (y_1, ..., y_n) ∈ I^n
        klíč π ∈ S_n
        optimální přiřazení parity (p, f)
VYSTUP:
        zpráva z = (z_1, ..., z_n) ∈ {0,1}^m
ALGORITMUS:
for 1, ..., m do
    z_i := p(y_π(i))
return z
```

Extrakčnímu algoritmu opět nezáleží na umístění barev v paletě. Má totiž k dispozici zobrazení p, které jednoznačně přiřadí barvě paritu. Tedy algoritmus se vždy podívá na index $y_{\pi(i)}$ odkazující do palety. Podívá se na kterou barvu odkazuje a tím pádem také ví, na jakou paritu odkazuje. Tuto paritu zapíše do bitu z_i.

Jako problém se může zdát, že na vstupu je délka zprávy. Ta se ale do obrázku nikam zvlášť neukládá. Odesílatel a příjemce se nejprve totiž dohodnou buď na pevné délce zpráv, nebo se dohodnou, že první (např. 16 bitů zprávy) bude informace o délce zprávy.

Nyní se podívejme na problém, jak získat jednoznačně p a f z palety barev. Optimální přiřazení parity je totiž obecně nejednoznačné.

- Nejednoznačnost p:
  - Můžeme převrátit parity barev v libovolné komponentě a máme opět optimální přiřazení parity.
- Nejednoznačnost f:
  - K jedné barvě můžou v paletě existovat dvě nejpodobnější barvy.

Jednoznačnost požadujeme, protože (p, f) se v rámci komunikace neposílá (odesílatel i příjemce si ho spočítají samostatně ze znalosti palety). Nejednoznačnosti se dá zbavit zavedením lineárního uspořádání na množině barev.

Barvy bychom mohli teoreticky mohli uspořádat podle indexu v paletě. To by ale bylo zranitelné vůči přeuspořádání palety. Barvy bychom mohli uspořádat také pomocí lexikografického uspořádání. Zde je ale problém s duplicitními barvami v paletě (to se může jevit jako blbost - proč dávat do palety jednu barvu vícekrát? Ale například standard GIF potřebuje počet barev v násobcích 2. Tedy pokud je barev nedostatek, tak dojde k zopakování některé z nich).

Řešením je uspořádat barvy lexikograficky a duplicitní barvy ztotožnit.

Algoritmus optimálního přiřazení parity

`VSTUP:`

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

`VYSTUP:`

optimální přiřazení parity (p, f) pro C

`ALGORITMUS:`

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

Popíšeme si tento algoritmus. Na vstupu se nachází paleta jakožto množina a tedy nám nezáleží na jejím uspořádání. V (1) se nachází množina E. Na tuto množinu se můžeme dívat jako na množinu hran v grafu. Ta obsahuje trojice: vzdálenost d mezi barvami (ta je stěžejní), c_i značí vrchol ze kterého vede hrana a c_j značí vrchol do kterého vede hrana. Množina E tak obsahuje hrany mezi libovolnými dvěma různými vrcholy v obou směrech. Jedná se tedy o úplný graf. V (3) vybíráme lexikograficky nejmenší trojici. Stěžejní je pro nás vzdálenost (chceme najít nejbližší barvy) a tedy se nejprve rozhodujeme podle ní. Pokud by se stalo, že nějaké dvojice mají stejnou vzdálenost, tak se podíváme na druhou (případně na třetí) složku. Toto nám zaručuje jednoznačnost.

Dále se nám algoritmus dělí na dvě větve. V první větvi nám vzniká tzv. „jádro“, tedy dva vrcholy mezi kterými vede navzájem cesta (tyto jádra můžeme vidět i Obrázku 17.).

Naším cílem je, aby ve výsledném „grafu“ (zobrazení f) byla vždy a pouze jediná hrana vedoucí z barvy do některé jiné. V kroku (11) proto dochází k odstranění veškerých hran z množiny E. Hledáme totiž nejbližší barvu k barvě c_i. Tu jsme už ale nalezli a cestu zaznačili v kroku (10) pomocí $f(i)$. Lepší cesta už není a tedy všechny cesty vedoucí z c_i můžeme z E odstranit (obdobně v kroku (7), zde ale vyhazujeme i cesty z c_j, protože jsme nalezli i nejkratší cestu z c_j $(f(j))$).

Důkaz správnosti algoritmu. Přiřazení $(d, c_i, c_j) := \min_{\mathtt{LEX}} E$ je dobře definované, protože paleta neobsahuje duplicity. Algoritmus nezávisí na uspořádání palety - nikde nedochází k porovnání indexů i a j. Algoritmus skončí, protože množina E je konečná a při každém průchodu cyklem se zmenší alespoň o jeden prvek (d, c_i, c_j) (minimálně vždy o ten, který zrovna v daném kole while cyklu řešíme).

Platí také, že po každém průchodu hlavním cyklem platí pro všechna $i \in I$ následující tři invarianty:

$$\begin{aligned} p(i) \text{ není definované} &\iff (E \bigcap (\mathbb{R} \times \{c_i\} \times C)) \neq \varnothing \\ p(i) \text{ je definované} &\iff f(i) \text{ je definované} \\ p(i) \text{ je definované} &\implies p(i) = 1 - p(f(i)). \end{aligned}$$

Na konci algoritmu tedy platí:

- $p(i) \in \{0, 1\}$ pro všechna $i \in I$. (Množina E je prázdná).
- f je řádně definované zobrazení.
- f obrací paritu.
- Pro každé $i \in I$ je $||c_i - c_{f(i)}|| = \min_{j \in I \setminus \{i\}} ||c_i - c_j||$.

Poznámka. Existují ještě modifikace vkládání. Například vkládání při redukci hloubky barev:

- Nosič je např. 24-bitový rastrový obrázek.
- Sestrojíme paletu a optimální přiřazení parity.
- Provedeme redukci hloubky barev s difúzí chyb.
- U pixelů nesoucích bit zprávy postup upravíme:
  - Při zaokrouhlování na nejbližší barvu se omezíme na množinu barev jejichž parita odpovídá bitu zprávy.

Další modifikací je tzv. Adaptivní vkládání
Nápad:

- Nechceme vkládat do oblastí s uniformní barvou. (Přesvícené oblasti, nebe a jiné gradienty).
- Chceme vkládat do oblastí s vysokou texturou.

Příklad:

- Obrázek rozdělíme na bloky 3×3 pixely.
- Definujeme funkci t, která měří texturu bloku.
- Např. $t(B)$ = počet různých barev v B.
- Zvolíme prahovou hodnotu γ, např. $\gamma = 2$, nebo $\gamma = 3$.
- Do bloku vkládáme, právě když $t(B) > \gamma$.

Problém:

- Vkládání může texturu snížit.
- Příjemce nemá k dispozici nosič, aby mohl určit co byla oblast s vysokou či nízkou texturou.

Řešení:

- Poté co provedeme vkládání do bloku, změříme texturu znovu.
- Klesla-li textura pod prahovou hodnotu:
  - Blok odešleme se změnami.
    - Kdybychom totiž blok odeslali bez změn, platilo by pro něj $t(B) > \gamma$ a příjemce by se z něj tedy snažil získat zprávu.
  - Blok bude příjemcem ignorován.
  - Vložená informace je ztracena.
  - Bity musíme vložit znovu do dalšího bloku.

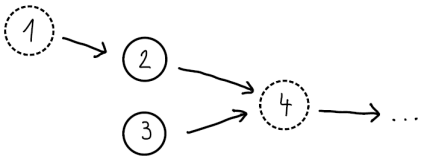

LSB s paletou setříděnou podle jasu
Vkládání s optimální paritou
Vkládání s optimální paritou a difuzí chyb
Nosič

5. Popište kvantitativní útok na vkládání s optimálním přiřazením parity.
Textový zdroj: naskenované poznámky - „LSB Embedding“ (POZOR! útok až z 14. strany - předchozí se nepřednášel)

Idea:

- Z histogramu stegoobrázku zrekonstruujeme histogram nosiče pro různé hodnoty α.
- Jestliže α zvolíme větší než je jeho skutečná hodnota, objeví se v zrekonstruovaném histogramu záporné hodnoty.
- Předpokládáme, že barvy v histogramu nemají rovnoměrné rozdělení - ještě lépe: předpokládáme, že se ve stegoobrázku nachází barva s vysokou četností, která sousedí (tedy z ní vede zobrazení f) s barvou s nízkou četností.

Nejprve se podívejme na ilustrační příklad. Předpokládejme že máme v obrázku následující situaci s barvami 1, 2, 3 a 4. Útočník obdrží obrázek, který obsahuje paletu. Tedy i on si může spočítat p a f.



Opět zavedeme značení pro histogramy $h(i)$ (počet výskytu indexu i v matici indexů ve stegoobrázku) a $h_0(i)$ (počet výskytu indexu i v matici indexů v nosiči). Připomeňme, že $\beta = \alpha/2$ (kde $\alpha = \frac{m}{n}$) značí pravděpodobnost, že došlo ke změně při vkládání zprávy do nosiče, tedy že došlo ke změně barvy z původní která byla v nosiči, aby tak došlo ke změně parity. Ze zobrazení f (tedy grafu na obrázku) můžeme zapsat následující rovnice.

$$\begin{aligned}
h(1) &\doteq (1-\beta)h_0(1)\\
h(2) &\doteq (1-\beta)h_0(2) + \beta h_0(1)\\
h(3) &\doteq (1-\beta)h_0(3)\\
h(4) &\doteq (1-\beta)h_0(4) + \beta(h_0(2) + h_0(3))
\end{aligned}$$

Například rovnice pro $h(2)$ vznikla tak, že s pravděpodobností β došlo ke změně v indexu 1 (to znamená, že se index 1 změnil na index 2) a tedy $h_0(1)$ (počet indexů 1 v nosiči) se změnil na index 2 s touto pravděpodobností. Zároveň s pravděpodobností $(1-\beta)$ nedošlo ke změně indexu 2 a tedy $(1-\beta)h_0(2)$ indexů 2 má i ve steganoobrázku hodnotu 2.

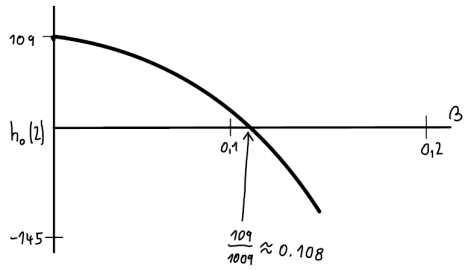
My ale máme k dispozici stegoobrázek a tedy známe hodnoty $h(1), \ldots, h(4)$. Co chceme určit jsou hodnoty $h_0(1), \ldots, h_0(4)$. Proto si je vyjádříme z rovnic (pro tento příklad bude stačit vyjádřit pouze $h_0(1)$ a $h_0(2)$).

$$\begin{aligned}
h_0^{(\beta)}(1) &\doteq \frac{1}{1-\beta}h(1)\\
h_0^{(\beta)}(2) &\doteq \frac{1}{1-\beta}(h(2) - \beta h_0(1))\\
&\doteq \frac{1}{1-\beta}\left(h(2) - \frac{\beta}{1-\beta}h(1)\right)
\end{aligned}$$

U h_0 píšeme v horním indexu (β), protože to jsou hodnoty závislé na hodnotě β, kterou neznáme.

Předpokládejme pro tento příklad, že máme $\alpha = 0.2 \implies \beta = 0.1$ a že $h_0(1) = 1000, h_0(2) = 10$. Představme si ideální situaci a tedy $h(1) \approx 1000 \cdot 0.9 = 900$ a $h(2) \approx 9 + 0.1 \cdot 10 = 9 + 100 = 109$.

Nyní předpokládejme, že jsme útočník. Ten je schopen z obrázku, který má k dispozici, určit $h(1)$ a $h(2)$ a tedy ví, že $h(1) = 900$ a $h(2) = 109$. Budeme chtít určit β. Podívejme se na následující graf.



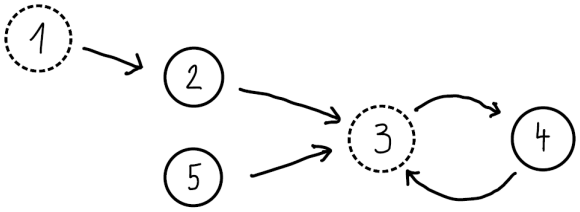
Z něj vidíme, že hodnota β je v intervalu $[0, \frac{109}{1009}]$ (ano, skutečně je pravá mez $\frac{109}{1009}$), což je přibližně $[0, 0.108]$. Získali jsme tak horní odhad pro β, který je velmi přesný (zde je přesnost důsledkem ideálního stavu $h(1)$ a $h(2)$ a taky, že rozdíl mezi $h_0(1)$ a $h_0(2)$ jsme zvolili tak velký).

Nyní se pokusíme tento postup zobecnit.

Definice. *Matici incidence* $A = (a_{ij}) \in \mathbb{R}^{|C|\times|C|}$ definujeme jako

$$a_{ij} = \begin{cases} 1 \text{ pokud } f(j) = i \\ 0 \text{ jinak} \end{cases}.$$

Tedy index sloupce určuje „z“ a řádek určuje „do“. Jinak řečeno, cestu z bodu 1 do 2 zaznačíme jedničkou na v prvním sloupci a druhém řádku. Tedy pro následující graf



je matice incidence

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

Zapišme nyní obecně vkládání. Označme si vektory hodnot histogramů ze stegoobrázku a nosiče

$$\overrightarrow{h} := \begin{pmatrix} h(1) \\ \vdots \\ h(|C|) \end{pmatrix}, \qquad \overrightarrow{h_0} = \begin{pmatrix} h_0(1) \\ \vdots \\ h_0(|C|) \end{pmatrix}.$$

Potom platí následující vztah

$$\overrightarrow{h} \approx (\beta \cdot A + (1-\beta)I)\overrightarrow{h_0}.$$

Ten vychází z toho, že do hodnoty $h(j)$ potřebujeme změřit, co nám do $h(j)$ „přiteče“ z ostatních vrcholů. To víme díky matici incidence a tedy, co se dostane do j z ostatních vrcholů je rovno $\beta \cdot A \cdot \overrightarrow{h_0}$. Také potřebujeme změřit, co ve vrcholu j zůstane, což je rovno $(1-\beta)I \cdot \overrightarrow{h_0} = (1-\beta)\overrightarrow{h_0}(j)$.

My ale opět $\overrightarrow{h}$ známe a chceme určit $\overrightarrow{h_0}$. Vyjádříme si tedy

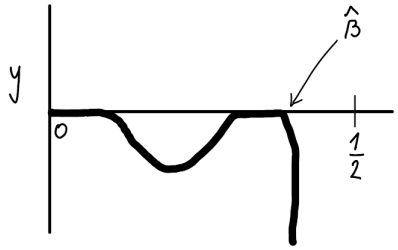
$$\overrightarrow{h_0}^{(\beta)} \doteq (\beta \cdot A + (1-\beta)I)^{-1}\overrightarrow{h}.$$

Teď se můžeme konečně podívat na útok. V něm $\forall \beta \in [0, \frac{1}{2}]$ zrekonstruujeme $\overrightarrow{h_0}$ a označme hodnotu y (závislou na β) jako

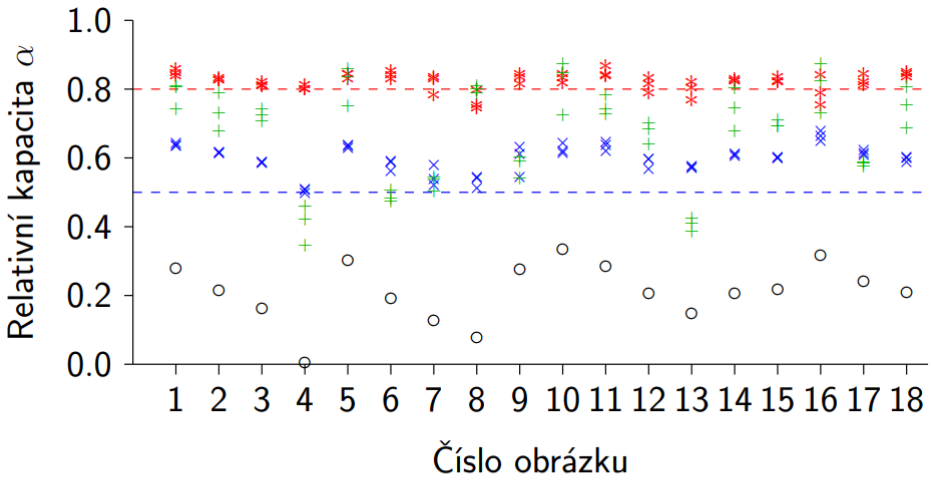
$$y(\beta) = \min_i h_0^{(\beta)}(i).$$

Za náš odhad $\hat{\beta}$ zvolíme maximální hodnotu v intervalu $[0, \frac{1}{2}]$, pro kterou $y(\beta)$ mění znaménko z + na - (tedy protíná osu x). Potom $\hat{\beta}$ je horní odhad pravděpodobnosti změny ($\hat{\alpha} = 2\hat{\beta}$).

Funkce y může mít například tento tvar:



Tento útok funguje dobře pro $\alpha < 0.5$. Když vypustíme barvy v jádrech, pak funguje dobře pro $\alpha > 0.5$.



Graf zobrazující odhad relativní kapacity pro nosiče (○) a stegoobrázky s $\alpha = 0.5$ (×), $\alpha = 0.8$ (*) a $\alpha = 1$ (+). Obrázky 512×512 s 256 barvami vytvořené pomocí ImageMagick z http://www4.comp.polyu.edu.hk/~cslzhang/CDM_Dataset.htm.

Poznámka. Podívejme se na následující myšlenkový experiment. Takový pohled na to, proč je dobré mít pro tento útok dva vrcholy takové, aby jeden měl velkou hodnotu a druhý, do nějž z toho s velkou hodnotou vede cesta, který má malou hodnotu.

Představme si, že máme nějakou soustavu vodních nádrží (body v grafu) a z nádrže vede vodopád do jiné právě tehdy, když je v grafu cesta. Máme tedy nějakou nádrž plnou vody (označme ji 1), ze které teče vodopád do jiné nádrže, ve které je málo vody (tu označme 2). Máme tedy velký rozdíl v množství vody v těchto nádržích. Vodopád který padá z první nádrže do druhé je mohutný (to koreluje s tím, že máme velké množství indexu 1 a tedy taky velké množství z těchto indexů se změní na index

2 (s pravděpodobností β)). Z nádrže 2 jistě teče vodopád do nějaké nádrže 3, ale tento vodopád je oproti vodopádu z 1 do 2 rozhodně menší.

Nyní si představme, že voda neteče, spustíme nahrávání na kameru a v ten moment začne voda přetékat. Po nějaké chvíli kameru zastavíme. Nyní se podíváme na záznam a poslední snímek z videa nám představuje v tomto modelu stegoobjekt - máme nějak popřelévané indexy, ale jaký byl původní stav? To můžeme zjistit jedině tak, že spustíme záznam v opačném směru.

Záznam už ale nemám k dispozici. Jediné co máme, je fotka koncového stavu. Můžeme tedy začít simulovat zpětný chod. Tedy voda se ve velkém vrací z nádrže 2 do nádrže 1. Zajisté se i nějaká voda vrací do nádrže 2, ale protože vodopád z 1 do 2 byl větší než z 2 do 3, tak po nějaké době dojde k úplnému vyprázdnění nádrže 2. V tento moment můžeme zpětný chod zastavit. Dostali jsme se totiž do situace, kdy by po spuštění nemohla téct voda. Tedy tento stav můžeme vzít jako krajní odhad a víme tak, že situace v nádržích se nachází někde v intervalu „spuštění zpětného chodu“ a „vyprázdnění nádrže 2“.

Přesně tohohle dosáhneme když se některá z hodnot $h_0(i)$ dostane na hodnotu 0. Kdybychom pokračovali ve zpětném chodu dál, byla by hodnota $h_0(i)$ záporné, což nejde (stejně tak jako nejde mít záporné množství vody).

6. Popište stegosystém OutGuess a odvoďte vzorce pro relativní kapacitu nosiče.
Textový zdroj: naskenované poznámky - „Principy tvorby stegosystému“

Cílem našeho snažení je vytvářet stegoobjekty, které by zachovávaly statistické vlastnosti nosičů. S tím je ale spojeno několik problémů: nemáme žádný jednoduchý a zároveň přesný statistický model pro digitální fotografie a digitální fotografie vykazují mnoho složitých lokálních vztahů kvůli povaze senzorů a kvůli dalšímu zpracování obrazu (viz. akvizice obrazu - 1. naskenovaný dokument).

Řešení:

(1) Stegosystém zachovávající nějaký zjednodušený statistický model nosiče.
(2) Stegosystém, který napodobuje nějaký přirozený proces, zpravidla šum. Problém:
   - Šum vzniká ještě předtím, než vůbec dojde ke konverzi zachyceného signálu na digitální signál, obraz potom prochází demosaikováním, korekcí barev a filtry. Na závěr může navíc projít ztrátovou kompresí.
(3) Stegosystém odolný proti stávajícím útokům (známým).
   - Řešením by mohlo být vkládat do malé části obrázku, nebo do oblasti s vysokou texturou.
   - V dnešní problém se k útoku používá neuronová síť u které vlastně nevíme, jak k výsledku dojde a tedy nevíme, jak se bránit.
(4) Minimalizovat dopad změn způsobených vkládáním.

Příklad stegosystému zachovávajícího model nosiče - OutGuess (vylepšený Jsteg).

- Modelujeme kvantizované DCT koeficienty v obrázku JPEG jako nezávislé náhodné veličiny se stejným rozdělením.
- Cíl: zachovat histogram

Algoritmus OutGuess (vylepšený Jsteg)

(1) Zaznamenáme histogram nosiče.
(2) Na základě stegoklíče zvolíme posloupnost DCT koeficientů do kterých budeme vkládat. Koeficienty s hodnotou 0 a 1 vynecháme (vkládání do 0 zanechává viditelné stopy). Nebudeme vkládat do celého obrázku, ale pouze do části a to proto, že potřebujeme mít nezměněné koeficienty, které poté budeme měnit tak, abychom histogram předělali do normálního tvaru. Horní odhad na α si ukážeme později v této otázce.
(3) V první fázi vkládáme do zvolených DCT koeficientů pomocí LSB embeddingu (tím poškodíme histogram).
(4) V druhé fázi obnovíme původní histogram tím, že koeficienty, do kterých jsme nevkládali využijeme ke korekci histogramu.

Do jak velkého podílu α nenulových koeficientů budeme vkládat? Musíme si totiž nechat prostor ke korekci.

Co se děje při vkládání? Podle nám již známého vztahu $x_i \in \{2k, 2k+1\} \iff y_i \in \{2k, 2k+1\}$ víme, že se ovlivňují pouze dva vedlejší sloupce $2k$ a $2k+1$. K přelití dojde s pravděpodobností $\beta = \frac{\alpha}{2}$. Čili změna v histogramu v rámci k-tého páru je

$$\left|\frac{\alpha}{2}h_0(2k+1) - \frac{\alpha}{2}h_0(2k)\right|.$$

Menší sloupec vzroste a větší poklesne o tento rozdíl. Pro úpravu histogramu je špatné, když jsou v páru vysoký a nízký sloupec.

Korekce tedy musí „přesunout“ nevyužité koeficienty z menšího sloupce do většího, abychom napravili změnu způsobenou vkládáním. Podíváme se na počet koeficientů použitelných pro korekci

$$(1-\alpha)\min(h_0(2k), h_0(2k+1)),$$

kde $(1-\alpha)$ vyjadřuje část koeficientů, které jsme vkládáním nezměnili a minimum vyjadřuje fakt, že problematičtější je menší sloupeček. Potřebujeme aby těchto použitelných koeficientů bylo dostatek, tedy aby jich bylo minimálně tolik, kolik se jich změní, tedy

$$(1-\alpha)\min(h_0(2k), h_0(2k+1)) \geq \frac{\alpha}{2}|h_0(2k+1) - h_0(2k)|.$$

Nyní se pokusíme vyjádřit z tohoto vztahu relativní kapacitu nosiče α. Vydělíme obě strany prvky α a minimem. Tím získáme

$$\frac{1-\alpha}{\alpha} \geq \frac{|h_0(2k+1) - h_0(2k)|}{2\min(h_0(2k), h_0(2k+1))}.$$

Platí, že $\frac{1-\alpha}{\alpha} = \frac{1}{\alpha} - 1$ a tedy

$$\frac{1}{\alpha} - 1 \geq \frac{|h_0(2k+1) - h_0(2k)|}{2\min(h_0(2k), h_0(2k+1))}.$$

Nyní převedeme -1 na pravou stranu a do zlomku.

$$\frac{1}{\alpha} \geq \frac{|h_0(2k+1) - h_0(2k)| + 2\min(h_0(2k), h_0(2k+1))}{2\min(h_0(2k), h_0(2k+1))}$$

Nyní dáme nerovnici na mocninu -1.

$$\alpha \leq \frac{2\min(h_0(2k), h_0(2k+1))}{|h_0(2k+1) - h_0(2k)| + 2\min(h_0(2k), h_0(2k+1))}$$

Podívejme se nyní na výraz ve jmenovateli. Ten se pokusíme zjednodušit. Mohou nastat pouze dva případy. Buď (a) $h_0(2k+1) < h_0(2k)$ a nebo (b) $h_0(2k) < h_0(2k+1)$.

V případě (a) potom $2\min(h_0(2k), h_0(2k+1)) = 2h_0(2k+1)$ a zároveň $|h_0(2k+1) - h_0(2k)| = h_0(2k) - h_0(2k+1)$ a tedy ve jmenovateli máme $h_0(2k) - h_0(2k+1) + 2h_0(2k+1) = h_0(2k) + h_0(2k+1)$.

V případě (b) potom $2\min(h_0(2k), h_0(2k+1)) = 2h_0(2k)$ a $|h_0(2k+1) - h_0(2k)| = h_0(2k+1) - h_0(2k)$ a tedy ve jmenovateli máme $h_0(2k+1) - h_0(2k) + 2h_0(2k) = h_0(2k+1) + h_0(2k)$.

Jak v případě (a) tak v případě (b) nám vyšly stejné hodnoty a tedy

$$\alpha \leq \frac{2\min(h_0(2k), h_0(2k+1))}{h_0(2k) + h_0(2k+1)},$$

což platí $\forall k \in \mathbb{Z} \setminus \{0\}$ (do indexů 0 a 1 zprávu nezapisujeme). Otázkou tedy je, pro které k je výraz

$$\frac{2\min(h_0(2k), h_0(2k+1))}{h_0(2k) + h_0(2k+1)}$$

minimální. Prvek 2 z čitatele můžeme zanedbat a otázku tak lze přeformulovat následovně: Pro které k je výraz

$$\frac{h_0(2k) + h_0(2k+1)}{\min(h_0(2k), h_0(2k+1))}$$

maximální. Tento výraz můžeme dále přeformulovat do výrazu

$$\frac{\max(h_0(2k), h_0(2k+1))}{\min(h_0(2k), h_0(2k+1))},$$

který se snažíme opět maximalizovat. Proč? Podívejme se na následující úpravy. Vycházíme z

$$\frac{h_0(2k) + h_0(2k+1)}{\min(h_0(2k), h_0(2k+1))}.$$

Od tohoto výrazu můžeme odečíst jedničku (tím se maximalizace nepokazí) a tedy

$$\frac{h_0(2k) + h_0(2k+1)}{\min(h_0(2k), h_0(2k+1))} - 1 = \frac{h_0(2k) + h_0(2k+1) - \min(h_0(2k), h_0(2k+1))}{\min(h_0(2k), h_0(2k+1))}.$$

Nyní se nám případ rozvětví na dvě možnosti

(a) $h_0(2k) < h_0(2k+1)$: To znamená, že máme

$$\frac{h_0(2k) + h_0(2k+1) - h_0(2k)}{h_0(2k)} = \frac{h_0(2k+1)}{h_0(2k)} = \frac{\max(h_0(2k), h_0(2k+1))}{\min(h_0(2k), h_0(2k+1))}$$

(b) $h_0(2k+1) < h_0(2k)$: To znamená, že máme

$$\frac{h_0(2k) + h_0(2k+1) - h_0(2k+1)}{h_0(2k+1)} = \frac{h_0(2k)}{h_0(2k+1)} = \frac{\max(h_0(2k), h_0(2k+1))}{\min(h_0(2k), h_0(2k+1))}$$

V obou případech jsme tedy získali, že se snažíme přes všechna k maximalizovat výraz

$$\frac{\max(h_0(2k), h_0(2k+1))}{\min(h_0(2k), h_0(2k+1))}.$$

To znamená, že hledáme v histogramu takové dva sloupce $\{2k, 2k+1\}$, mezi kterými je největší rozdíl ve velikostech, tedy chceme, aby $|h_0(2k) - h_0(2k+1)|$ bylo co největší.

Ve fotografiích se zpravidla jedná o pár $\{2k, 2k+1\} = \{-2, -1\}$, čili máme

$$\alpha_{\max} = \frac{2h_0(-2)}{h_0(-1) + h_0(-2)}.$$

Obvykle $\alpha_{\max} = 0.2$ bpnc (bits per non-zero coefficient).

7. Popište algoritmus vkládání využívaný ve stegosystému F5 a odvoďte vzorec pro relativní kapacitu nosiče bez maticového kódování.
Textový zdroj: naskenované poznámky - „Principy tvorby stegosystému“, Fridrich J. - „Steganography in Digital media: Principles, algorithms and applications“ strana 121 (Algorithm 7.1)

Začněme úvodem, jak bychom mohli vylepšit Jsteg. Zprávu budeme opět ukládat do LSB v DCT koeficientech. Nově budeme zprávu vkládat pouze do nenulových AC koeficientů (tj. včetně 1) a to takto:

- Má-li koeficient správný LSB, pak ho neměníme.
- V opačném případě dekrementujeme absolutní hodnotu koeficientu o 1 (tedy například. z 3 se stane 2 a z -3 se stane -2).

Tímto nám ale vzniká nový problém. Vložíme-li do koeficientu -1 nebo 1 hodnotu 0, pak výsledkem bude koeficient 0. Ale tento se při extrakci zprávy nečte. Tomuto fenoménu se říká „skrinkage“ = „smrštění“. Řešením je v případě smrštění vložit bit znovu.

K tomu se váže ale další problém. Smrštění nastává pouze při vkládání bitu 0 (do DCT koeficientů 1 nebo -1). Opětovné vkládání nulových bitů znamená, že celkově vkládáme více nul než jedniček.

Podívejme se, jak moc nul oproti jedničkám bychom takovým to způsobem vkládali. Označme

- $P(Z_i)$ pravděpodobnost, že v i-tém kroku vkládáme bit 0
- $P(S_i)$ pravděpodobnost, že v i-tém kroku dojde ke smrštění

Vezměme si situaci, že v i-tém kroku vkládáme 0. Co tomu předcházelo? Jsou dvě varianty. První možností je, že v $(i-1)$-tém kroku došlo ke smrštění. To znamená, že jsme v $(i-1)$-ním kroku vkládali bit 0 do koeficientu 1 nebo -1 a tedy je potřeba bit 0 vložit znovu. Druhou možností je, že ke smrštění v předcházejícím kroku nedošlo a my tedy pouze vkládáme 0.

Všimněme si zároveň, že $P(Z_0) = \frac{1}{2}$ protože ke smrštění nemohlo v předcházejícím kroku dojít (žádný nebyl, jsme v kroku 0) a zpráva má rovnoměrné rozdělení a tedy pravděpodobnost, že vkládáme bit 0 je 0.5.

Máme tak

$$P(Z_i) = P(S_{i-1}) + \frac{1}{2}(1 - P(S_{i-1})),$$

kde $(1 - P(S_{i-1}))$ je pravděpodobnost, že v $(i-1)$-ním kroku nedošlo ke smrštění a $\frac{1}{2}$ kvůli tomu, že $z_{i-1} = 0$ s pravděpodobností 0.5.

Dále máme

$$P(S_i) = P(Z_i)\frac{h(-1) + h(1)}{\sum_{i\neq 0} h(i)},$$

kde h je histogram AC koeficientů. $P(Z_i)$ se zde vyskytuje, protože ke smrštění dojde pouze, když vkládáme 0. V čitateli sčítáme $h(-1)$ a $h(1)$, protože ke smrštění dojde pouze pokud vkládáme do 1 nebo -1. Ve jmenovateli dělíme počtem všech nenulových AC koeficientů.

Dosaďme nyní za $P(S_{i-1})$ do rovnice pro $P(Z_i)$.

$$P(Z_i) = \frac{1}{2}P(Z_{i-1})\frac{h(-1) + h(1)}{\sum_{i\neq 0} h(i)} + \frac{1}{2}$$

Nás by nyní zajímalo, jak se bude pravděpodobnost $P(Z_i)$ vyvíjet pro rostoucí i, tedy nás zajímá

$$\lim_{i\to\infty} P(Z_i).$$

Předpokládejme, že tato limita posloupnosti $P(Z_i)$ existuje. Označme ji A. Potom z definice limity posloupnosti máme

$$\forall \epsilon \in \mathbb{R}, \epsilon > 0 \ \exists i_0 \in \mathbb{N} \ \ \forall i \in \mathbb{N}, i \geq i_0 : |P(Z_i) - A| < \epsilon.$$

Z toho plyne, že pro libovolné $\epsilon' > 0$ existuje i_0 takové, že pro $i - 1 \geq i_0$ platí $|P(Z_{i-1}) - P(Z_i)| < \epsilon'$ a tedy můžeme uvažovat $P(Z_{i-1}) \approx P(Z_i)$.

Potom

$$P(Z_i) = \frac{1}{2}P(Z_i)\frac{h(-1)+h(1)}{\sum_{i\neq 0}h(i)} + \frac{1}{2}$$
$$P(Z_i)\left(1 - \frac{1}{2}\frac{h(-1)+h(1)}{\sum_{i\neq 0}h(i)}\right) = \frac{1}{2}$$
$$P(Z_i)\left(2 - \frac{h(-1)+h(1)}{\sum_{i\neq 0}h(i)}\right) = 1$$
$$P(Z_i) = \left(2 - \frac{h(-1)+h(1)}{\sum_{i\neq 0}h(i)}\right)^{-1}$$

a tedy

$$\lim_{i\to\infty} P(Z_i) = \lim_{i\to\infty}\left(\left(2 - \frac{h(-1)+h(1)}{\sum_{i\neq 0}h(i)}\right)^{-1}\right).$$

Typicky platí

$$\frac{h(-1)+h(1)}{\sum_{i\neq 0}h(i)} \approx 0.5$$

a tedy pravděpodobnost, že vkládáme nulu $P(Z_i) \approx \frac{2}{3}$ a pravděpodobnost, že vkládáme 1 je $1-P(Z_i) \approx \frac{1}{3}$. Z toho plyne, že 0 vkládáme 2× více než 1!

Toto má za důsledek, že u lichých koeficientů je pravděpodobnost změny 2× větší než u sudých. Ovšem i tento problém má své řešení. Předefinujeme pojem LSB pro záporná čísla

$$\mathrm{LSB}_{\mathrm{F5}}(x) = \begin{cases}(1-x) \bmod 2 \text{ pro } x < 0\\ x \bmod 2 \text{ jinak}\end{cases}$$

Tato definice nám tedy říká, že pro kladná x se nic nemění. Ovšem pro záporná chápeme sudost/lichost naopak a tedy poslední bit v x chápeme pro záporná čísla obráceně.

Toto má za následek, že pokud chceme vkládat do DCT koeficientu –1 bit 0, tak nic měnit nemusíme a pokud chceme vložit bit 1, tak změníme DCT koeficient na 1. Tedy vložení 0 do 1 smršťuje a vložení bitu 1 do -1 smršťuje. Jelikož $h(1) \approx h(-1)$, tak dochází ke smršťování při vkládání 0 a 1 se stejnou pravděpodobností.

Relativní kapacita nosiče je v tomto modifikovaném případě rovna

$$1 - \frac{1}{2}\frac{h(-1)+h(1)}{\sum_{i\neq 0}h(i)} \approx 0.75 \text{ bpnc}$$

(bits per non-zero AC coefficient), protože o kapacit přijdeme jen tehdy když dojde ke smrštění, což nastane když dosazujeme do koeficientů 1 a -1. U koeficientů 1 dojde ke smrštění při dosazení 0 a u koeficientu –1 dojde ke smrštění při dosazení 1. Tedy pravděpodobnost, že budeme dosazovat bit, který způsobí smrštění je 0.5.

Nyní se podívejme na samotný algoritmus F5 (bez maticového vkládání). Ten využívá právě $\mathrm{LSB}_{\mathrm{F5}}$. Algoritmus není asi potřeba přesně znát, stačí jen vědět, co se v něm děje.

```
VSTUP:
        zpráva z = (z_1, ..., z_m) ∈ {0,1}^m
        nosič (DCT koeficienty) x = (x_1, ..., x_n) ∈ [-1024, 1024]^n
        klíč π ∈ S_n
VYSTUP:
        stegoobjekt y = (y_1, ..., y_m) ∈ [-1024, 1024]^n
ALGORITMUS:
(1)   y := x
(2)   i := 1, j := 1          % i je index ve zprávě; j je index DCT koeficientu
(3)   while(i ≤ m) & (j ≤ n) do
(4)       if (x_π(j) ≠ 0) & (x_π(j) is not DC term) then
(5)           if LSB_F5(x_π(j) = z_i) then
(6)               i := i + 1
(7)           else
(8)               y_π(j) := (x_π(j) - sign(x_π(j)))
(9)               if y_π(j) ≠ 0 then
(10)                  i := i + 1
(11)      j := j + 1
(12)  return y
```

Principem tedy je. Chci vložit z_i do $x_{\pi(j)}$. Vložení může proběhnout pouze, když $x_{\pi(j)}$ není nulový DCT koeficent a pokud $x_{\pi(j)}$ není DC koeficient.

Podíváme se na poslední bit v $x_{\pi(j)}$ pomocí funkce $\mathrm{LSB}_{\mathrm{F5}}$. Pokud $\mathrm{LSB}_{\mathrm{F5}}(x_{\pi(j)}) = z_i$, tak jsme hotovi, nemusíme nic měnit a můžeme jít na další bit zprávy. Pokud se však liší, je potřeba provést změnu.

V (8) probíhá ona dekrementace v závislosti na znaménku $x_{\pi(j)}$. Pokud je znaménko +, potom $y_{\pi(j)} = (x_{\pi(j)} - 1)$ a pokud je znaménko -, tak $y_{\pi(j)} = (x_{\pi(j)} + 1)$.

8. Formulujte a dokažte algoritmus maticového vkládání pomocí minimum-distance dekodéru.
Textový zdroj: naskenované poznámky - „Úvod do maticového vkládání“ (jedná se pouze o úvod), prezentace - „Steganografie pomocí maticového vkládání“, skripta - kapitola 2.3 (Algoritmus 2.3)

Pro maticové vkládání je potřeba trochu předefinovat již zavedené pojmy.

- Stegoobjekt rozdělíme na bloky délky n.
- Hovoříme o posloupnosti hodnot spjatých s blokem
  - nosiče $\mathbf{x} = (x_1, \dots, x_n) \in \mathbb{F}_q^n$ a
  - stegoobjektu $\mathbf{y} = (y_1, \dots, y_n) \in \mathbb{F}_q^n$
- Zprávu rozdělíme na bloky délky m.
- Předpokládáme, že „počet bloků nosiče“=„počet bloků zprávy“.
- Blok zprávy značíme $\mathbf{z} = (z_1, \dots, z_m) \in \mathbb{F}_q^m$.

Pro $q = 2$ jsou spjaté hodnoty například:

- LSB příslušného prvku nosiče či stegoobjektu.
- Parita ve smyslu vkládání s optimálním přiřazením parity v paletových formátech.

Tedy změnou je, že $\mathbf{x}$ už nechápeme jako vektor indexů z matice obrázku, ale už přímo jako jako například poslední bity.

Nyní připomeňme pár základních pojmů ze základního kurzu samoopravných kódů.

Definice. Podprostor $\mathcal{C}$ prostoru $\mathbb{F}_q^n$ nazýváme *lineární kód* délky n.

Dimenzi $\mathcal{C}$ značíme k.

Definice. *Hammingova váha* $w(\mathbf{u})$ vektoru $\mathbf{u} \in \mathbb{F}_q^n$ je počet nenulových složek v $\mathbf{u}$.

Definice. *Hammingova vzdálenost* vektorů $\mathbf{u}, \mathbf{v} \in \mathbb{F}_q^n$ je

$$d_H(\mathbf{u}, \mathbf{v}) := w(\mathbf{v} - \mathbf{u})$$

a Hammingova vzdálenost vektoru $\mathbf{u} \in \mathbb{F}_q^n$ od množiny $\mathcal{X} \subseteq \mathbb{F}_q^n$ je

$$d_H(\mathbf{u}, \mathcal{X}) = \min_{\mathbf{v} \in \mathcal{X}} w(\mathbf{v} - \mathbf{u}).$$

Definice. *Minimální vzdálenost* neboli *minimální váha* kódu $\mathcal{C}$ je $\min_{\mathbf{c} \in \mathcal{C} \setminus \{0\}} w(\mathbf{c})$.

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.

Definice. Matici $\mathbf{H}$ typu $(n - k) \times n$ nad tělesem $\mathbb{F}_q$ s lineárně nezávislými řádky nazýváme *paritní maticí* kódu $\mathcal{C}$, jestliže

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

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

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

Mějme rovnici $\mathbf{Hx} = \mathbf{s}$, potom říkáme, že $\mathbf{s}$ je syndromem vektoru $\mathbf{x}$. Rozkladová třída $\mathcal{C}(\mathbf{s})$ nám rozkládá prostor $\mathbb{F}_q^n$ na „rovnoběžné roviny“ s „rovinou“ kódu $\mathcal{C}$. Platí tedy $\bigcup_{\mathbf{s} \in \mathbb{F}_q^{n-k}} = \mathbb{F}_q^n$.

Nyní se podívejme na definici maticového vkládání a extrakce.

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

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

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

Zobrazení e z definice je pomocným technickým zobrazením, které vezme nějaký syndrom a najde k němu příslušný vektor. Můžeme ho chápat tak, že najde nějaké partikulární řešení $\mathbf{v}$ soustavy $\mathbf{Hv}=\mathbf{s}$. Toto řešení jistě leží v $\mathcal{C}(\mathbf{s})$, ovšem zobrazení e vezme z definice jedno z těch minimálních (může jich být více).

Vidíme, že

$$\mathrm{Ext}_{\mathbf{H}}(\mathrm{Emb}_{\mathbf{H},e}(\mathbf{x},\mathbf{z})) = \mathbf{H}(\mathbf{x}+e(\mathbf{z}-\mathbf{Hx})) = \mathbf{Hx}+\mathbf{z}-\mathbf{Hx} = \mathbf{z}$$

pro každé $\mathbf{x}\in\mathbb{F}_q^n$ a $\mathbf{z}\in\mathbb{F}_q^{n-k}$. Místo $\mathrm{Emb}_{\mathbf{H},e}$ budeme psát jen $\mathrm{Emb}_{\mathbf{H}}$, protože nás zobrazení e samo o sobě obvykle nezajímá. Podstatné je pouze to, že zobrazení 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 $\mathbf{u}\in\mathbb{F}_q^n$ platí $w(D(\mathbf{u})-\mathbf{u}) = d_H(\mathbf{u},\mathcal{C})$.

V případě Hammingových kódů je výpočet $e(\mathbf{s})$ jednoduchý. U jiných kódů už nemusí být výpočet $e(\mathbf{s})$ tak snadný, a v případě obecných kódů se dokonce jedná o NP-úplný problém. Následující algoritmus nám dává návod, jak spočítat $\mathrm{Emb}_{\mathbf{H}}$ pomocí minimum-distance dekodéru samoopravného kódu.

Algoritmus maticového vkládání s použitím minimum-distance dekodéru:

`VSTUP:`

nosič $\mathbf{x}\in\mathbb{F}_q^n$

zpráva $\mathbf{z}\in\mathbb{F}_q^{n-k}$

paritní matice $\mathbf{H}[n,k]_q$ kódu $\mathcal{C}$

minimum-distance dekodér D kódu $\mathcal{C}$

`VYSTUP:`

$\mathrm{Emb}_{\mathbf{H}}(\mathbf{x},\mathbf{z})$

`ALGORITMUS:`

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

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

Potřebujeme dokázat, že algoritmus má na výstupu skutečně $\mathrm{Emb}_{\mathbf{H}}(\mathbf{x},\mathbf{z})$ odpovídající definici maticového vkládání.

Začněme nejprve ověřením, že extrakcí dostaneme původní zprávu $\mathbf{z}$.

$$\mathrm{Ext}_{\mathbf{H}}(D(\mathbf{x}-\mathbf{u})+\mathbf{u}) = \mathbf{H}(D(\mathbf{x}-\mathbf{u})+\mathbf{u}) = \mathbf{H}(D(\mathbf{x}-\mathbf{u}))+\mathbf{Hu} = 0+\mathbf{z} = \mathbf{z}$$

Nula za druhou rovností máme díky tomu, že výstupem dekodéru jsou pouze kódová slova a pro kódové slovo $\mathbf{c}\in\mathcal{C}$ platí $\mathbf{Hc}=\mathbf{0}$.

Nyní ověříme, že výstup algoritmu splňuje definici maticového vkládání. Na výstupu by podle definice měl být výraz $\mathrm{Emb}_{\mathbf{H},e}(\mathbf{x},\mathbf{z}) = \mathbf{x}+e(\mathbf{z}-\mathbf{Hx})$. Tedy potřebujeme ověřit, zda můžeme volit za zobrazení $e(\mathbf{z}-\mathbf{Hx})$ výraz $D(\mathbf{x}-\mathbf{u})+\mathbf{u}-\mathbf{x}$.

Z definici maticového vkládání víme, že zobrazení e splňuje obecně dvě podmínky a to: pro všechna $\mathbf{s}\in\mathbb{F}_q^{n-k}$ platí $\mathbf{H}e(\mathbf{s})=\mathbf{s}$ a $w(e(\mathbf{s})) = \min_{\mathbf{u}\in\mathcal{C}(\mathbf{s})} w(\mathbf{u})$. Ověříme tedy tyto dvě podmínky pro výraz $D(\mathbf{x}-\mathbf{u})+\mathbf{u}-\mathbf{x} = D(\mathbf{x}-\mathbf{u})-(\mathbf{x}-\mathbf{u})$.

Zobrazení e musí v našem konkrétním případě splňovat první podmínku následovně

$$\mathbf{H}e(\mathbf{z}-\mathbf{Hx}) = \mathbf{z}-\mathbf{Hx}$$

a ověříme přímým výpočtem, že tato vlastnost je splněna i pro $D(\mathbf{x}-\mathbf{u})-(\mathbf{x}-\mathbf{u})$.

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

Nyní druhá podmínka. Pro zobrazení e musí v našem konkrétním případě platit

$$w(e(\mathbf{z}-\mathbf{Hx})) = \min_{\mathbf{v}\in\mathcal{C}(\mathbf{z}-\mathbf{Hx})} w(\mathbf{v}).$$

Zvolme tedy za $e(\mathbf{z}-\mathbf{Hx})$ náš výraz $D(\mathbf{x}-\mathbf{u})-(\mathbf{x}-\mathbf{u})$

$$
\begin{aligned}
(1) \qquad w(D(\mathbf{x}-\mathbf{u})-(\mathbf{x}-\mathbf{u})) &= d_H(\mathbf{x}-\mathbf{u},\mathcal{C}) \\
(2) \qquad &= \min_{\mathbf{c}\in\mathcal{C}} w(\mathbf{c}-(\mathbf{x}-\mathbf{u})) \\
(3) \qquad &= \min_{\mathbf{v}\in\mathcal{C}+\mathbf{u}-\mathbf{x}} w(\mathbf{v}) \\
(4) \qquad &= \min_{\mathbf{v}\in\mathcal{C}(\mathbf{H}(\mathbf{u}-\mathbf{x}))} w(\mathbf{v}) \\
(5) \qquad &= \min_{\mathbf{v}\in\mathcal{C}(\mathbf{z}-\mathbf{Hx})} w(\mathbf{v})
\end{aligned}
$$

Obě podmínky tak máme splněny a tedy algoritmus maticového vkládání s použitím minimum-distance dekodéru je korektní.

Pro úplnost si ještě popišme některé rovnosti z výpočtu váhy. Podívejme se na rovnost (1). Z definice minimum-distance dekodéru víme, že $D(\mathbf{x}-\mathbf{u})$ najde nejbližší kódové slovo k $\mathbf{x}-\mathbf{u}$, tedy váha je rovna vzdálenosti $\mathbf{x}-\mathbf{u}$ od kódu $\mathcal{C}$. Rovnost (2) máme z definice vzdálenosti od množiny.

Nyní se podívejme na rovnost (4). Nejprve máme $v \in \mathcal{C}+\mathbf{u}-\mathbf{x}$. Kód $\mathcal{C}$ je množina všech $\mathbf{c} \in \mathbb{F}_q^n$ takových, že $\mathbf{Hc}=\mathbf{0}$. Platí tedy, že $\mathbf{v}$ je tvaru $\mathbf{v}=\mathbf{c}+\mathbf{u}-\mathbf{x}$ pro každé $\mathbf{c} \in \mathcal{C}$. Tuto rovnost můžeme vynásobit maticí $\mathbf{H}$ a získáme tak

$$\mathbf{Hv}=\mathbf{Hc}+\mathbf{H}(\mathbf{u}-\mathbf{x}).$$

Víme, že $\mathbf{Hc}=\mathbf{0}$ a tedy

$$\mathbf{Hv}=\mathbf{H}(\mathbf{u}-\mathbf{x}).$$

Z této rovnice vidíme, že hledáme všechny $\mathbf{v} \in \mathbb{F}_q^n$ splňující tuto rovnost. To lze množinově vyjádřit jako

$$\{\mathbf{v} \in \mathbb{F}_q^n \mid \mathbf{Hv}=\mathbf{H}(\mathbf{u}-\mathbf{x})\},$$

ale toto je přesně definice $\mathcal{C}(\mathbf{H}(\mathbf{u}\text{-}\mathbf{x}))$.

9. Formulujte a dokažte větu o maticovém vkládání a její důsledek týkající se efektivity maticového vkládání.
Textový zdroj: skripta - kapitola 2.3 (Věta 2.4, Důsledek 2.5)

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

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

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

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

Pokrývací poloměr je vlastně minimální poloměr koule, kterou musíme nafouknout kolem každého slova z tohoto kódu, aby nám sjednocení těchto koulí pokrylo celý prostor.

Věta (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*

$$\max_{\boldsymbol{x},\boldsymbol{z}} d_H(\boldsymbol{x},\mathrm{Emb}_{\boldsymbol{H}}(\boldsymbol{x},\boldsymbol{z})) = r_c \qquad a \qquad \mathrm{E}_{\boldsymbol{x},\boldsymbol{z}}[d_H(\boldsymbol{x},\mathrm{Emb}_{\boldsymbol{H}}(\boldsymbol{x},\boldsymbol{z}))] = r_a,$$

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

Maximum v prvním vztahu vyjadřuje maximální velikost změny, tedy maximální vzdálenost nosiče od stegoobjektu při vkládání. Druhý vztah potom vyjadřuje střední vzdálenost Hammingovy vzdálenosti nosiče od stegoobjektu.

Důkaz. Nechť $\mathbf{s}\in\mathbb{F}_q^{n-k}$ je syndrom. Pak pro každé $\mathbf{u}\in\mathcal{C}(\mathbf{s})$ platí $d_H(\mathbf{u},\mathcal{C}) = w(e(\mathbf{s}))$, neboť

$$\begin{aligned}
&(1) \qquad d_h(\mathbf{u},\mathcal{C}) = \min_{\mathbf{c}\in\mathcal{C}} w(\mathbf{u}-\mathbf{c})\\
&(2) \qquad\qquad\quad = \min_{\mathbf{v}\in\mathcal{C}+\mathbf{u}} w(\mathbf{v})\\
&(3) \qquad\qquad\quad = \min_{\mathbf{v}\in\mathcal{C}(\mathbf{s})} w(\mathbf{v})\\
&(4) \qquad\qquad\quad = w(e(\mathbf{s}))
\end{aligned}$$

Rovnost (1) je čistě definice Hammingovy vzdálenosti od množiny. U rovnosti (2) bychom mohli psát $\mathbf{v}\in\mathbf{u}-\mathcal{C}$, ale $\mathcal{C}$ je vektorový prostor a tedy $\mathcal{C} = -\mathcal{C}$ a z toho můžeme psát $\mathbf{v}\in\mathcal{C}+\mathbf{u}$.

U rovnosti (3) využijeme podobný postup jako v otázce 8. rovnost (6). Tedy pro $\mathbf{v}$ platí, že je ve tvaru $\mathbf{v} = \mathbf{c}+\mathbf{u}$ pro všechna $\mathbf{c}\in\mathcal{C}$. Z toho plyne po vynásobení maticí $\mathbf{H}$, že $\mathbf{Hv} = \mathbf{Hc}+\mathbf{Hu} = \mathbf{Hu}$. My ale zároveň víme z $\mathbf{u}\in\mathcal{C}(\mathbf{s})$ že pro $\mathbf{u}$ platí $\mathbf{Hu} = \mathbf{s}$ a tedy $\mathbf{Hv} = \mathbf{s}$ což odpovídá $\mathbf{v}\in\mathcal{C}(\mathbf{s})$.

Rovnost (4) je potom z vlastnosti zobrazení e (vlastnosti jsou uvedeny v definici maticového vkládání a extrakce: pro všechna $\mathbf{s}\in\mathbb{F}_q^{n-k}$ platí $\mathbf{H}e(\mathbf{s}) = \mathbf{s}$ a $w(e(\mathbf{s})) = \min_{\mathbf{u}\in\mathcal{C}(\mathbf{s})} w(\mathbf{u})$).

Nyní se podívejme na první vztah, který chceme dokázat. Nechť $\mathbf{x}\in\mathbb{F}_q^n$, potom

$$\begin{aligned}
&(1) \qquad \max_{\mathbf{z}\in\mathbb{F}_q^{n-k}} d_H(\mathbf{x},\mathrm{Emb}_{\mathbf{H}}(\mathbf{x},\mathbf{z})) = \max_{\mathbf{z}\in\mathbb{F}_q^{n-k}} w(e(\mathbf{z}-\mathbf{Hx}))\\
&(2) \qquad\qquad\qquad\qquad\quad = \max_{\mathbf{s}\in\mathbb{F}_q^{n-k}} w(e(\mathbf{s}))\\
&(3) \qquad\qquad\qquad\qquad\quad = \max_{\mathbf{s}\in\mathbb{F}_q^{n-k}} \max_{\mathbf{u}\in\mathcal{C}(\mathbf{s})} d_H(\mathbf{u},\mathcal{C})\\
&(4) \qquad\qquad\qquad\qquad\quad = \max_{\mathbf{u}\in\mathbb{F}_q^n} d_H(\mathbf{u},\mathcal{C})\\
&(5) \qquad\qquad\qquad\qquad\quad = r_c
\end{aligned}$$

V rovnosti (1) jsme pouze využili následující. Z definice maticového vkládání víme, že $\mathrm{Emb}_{\mathbf{H}}(\mathbf{x},\mathbf{z}) = \mathbf{x}+e(\mathbf{z}-\mathbf{Hx})$ a z definice Hammingovy vzdálenosti dostáváme

$$d_H(\mathbf{x},\mathrm{Emb}_{\mathbf{H}}(\mathbf{x},\mathbf{z})) = d_H(\mathbf{x},\mathbf{x}+e(\mathbf{z}-\mathbf{Hx})) = w(\mathbf{x}+e(\mathbf{z}-\mathbf{Hx})-\mathbf{x}) = w(e(\mathbf{z}-\mathbf{Hx})).$$

Rovnost (2) máme díky tomu, že jsem si zvolili nějaké jedno $\mathbf{x}$ a tedy $\mathbf{Hx}$ je pro nás konstantní. Tedy jak pro $\mathbf{z}$ tak pro $\mathbf{s}$ procházíme celé $\mathbb{F}_q^{n-k}$, pokaždé ovšem v jiném pořadí.

V rovnosti (3) využíváme odvozený vztah na začátku důkazu. Zvětšíme množinu, kterou v maximu procházíme, ale díky dokázané pomocné rovnosti víme, že rovnost $d_H(\mathbf{u},\mathcal{C}) = w(e(\mathbf{s}))$ platí pro každé $\mathbf{u} \in \mathcal{C}(\mathbf{s})$ a tedy procházením více prvků nenajdeme nějaké nové maximum.

V rovnosti (4) využíváme toho, že $\mathcal{C}(\mathbf{s})$ jsou rozkladové třídy. Tedy díky $s \in \mathbb{F}_q^{n-k}$ se dostaneme do každé rozkladové třídy a díky $\mathbf{u} \in \mathcal{C}(\mathbf{s})$ projdeme každý prvek v této rozkladové třídy. Rozkladové třídy jsou navzájem disjunktní a tvoří celý prostor $\mathbb{F}_q^n$. Procházíme tedy všechny prvky $\mathbb{F}_q^n$. (Rozkladové třídy si můžeme (sice nepřesně) představit jako rovnoběžné roviny s rovinou $\mathcal{C}$.)

Poslední rovnost je už pouze definice r_c.

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

(1) $$\mathrm{E}_{\mathbf{x},\mathbf{z}}[d_H(\mathbf{x}, \mathrm{Emb}_{\mathbf{H}}(\mathbf{x},\mathbf{z}))] = \mathrm{E}_{\mathbf{s}}[w(e(\mathbf{s}))]$$

(2) $$= \frac{1}{q^{n-k}} \sum_{\mathbf{s}\in\mathbb{F}_q^{n-k}} w(e(\mathbf{s}))$$

(3) $$= \frac{1}{q^{n}} \sum_{\mathbf{s}\in\mathbb{F}_q^{n-k}} q^k w(e(\mathbf{s}))$$

(4) $$= \frac{1}{q^{n}} \sum_{\mathbf{s}\in\mathbb{F}^{n-k}} \sum_{\mathbf{u}\in\mathcal{C}(\mathbf{s})} d_H(\mathbf{u},\mathcal{C})$$

(5) $$= \frac{1}{q^{n}} \sum_{\mathbf{u}\in\mathbb{F}_q^{n}} d_H(\mathbf{u},\mathcal{C})$$

(6) $$= r_a$$

Rovnost (1) máme s použitím prvních dvou rovností v důkazu předchozího výrazu.

V rovnosti (2) využíváme toho, že $\mathbf{z}$ má rovnoměrné rozdělení a tedy i $\mathbf{s}$.

V rovnosti (3) jsme pouze přesunuli q^{-k} do výrazu.

Pro rovnost (4) využijeme opět rovnost z úvodu důkazu. V $d_H(\mathbf{u},\mathcal{C})$ se nám objeví $\mathbf{u}$, které může být úplně libovolné z $\mathcal{C}(\mathbf{s})$, ale které si vybrat? Použijeme trik, že projdeme všechny takové $\mathbf{u}$. Zároveň víme, že $|\mathcal{C}| = q^k$. Protože $\mathcal{C}(\mathbf{s})$ je jen „posunutá rovina“ a jedná se o rozkladovou třídu, tak také $|\mathcal{C}(\mathbf{s})| = q^k$. Tedy celkově procházením $\mathbf{u} \in \mathcal{C}(\mathbf{s})$ projdeme právě q^k možností, která každá dá stejný výsledek $d_H(\mathbf{u},\mathcal{C})$.

V rovnosti (4) využijeme stejný argument jako v rovnosti (4) u předcházejícího výrazu.

Poslední rovnost je už pouze definice r_a. □

Důsledek. *Jestliže $q \in \{2,3\}$, pak efektivita maticového vkládání pro $[n,k]_q$ kód $\mathcal{C}$ je*

$$e = (n-k)(\log_2 q)/r_a,$$

kde r_a je průměrná vzdálenost od kódu.

Délka zprávy v bitech je $(n-k)\log_2(q)$, protože n je počet sloupců paritní matice a k je pořet řádků generující matice, tak $n-k$ je počet řádků paritní matice, což odpovídá délce zprávy kterou můžeme vložit do jednoho bloku délky n. Výraz $\log_2(q)$ je počet bitů, které vložíme na jeden prvek bloku. Celý výraz dělíme průměrnou distorzí, kterou uděláme a tím, že tyto změny jsou maximálně jedna, tak v tomto případě r_a=distorze.

10. Odvoďte spodní mez na pokrývací poloměr v závislosti na relativní kapacitě.
Textový zdroj: skripta - kapitola 2.4 (Lemma 2.6, Věta 2.7)

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

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

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

Definice. Pro $\mathbf{v} \in \mathbb{F}_q^n$ a $r \geq 0$, definujeme *kouli se středem* $\mathbf{v}$ a poloměrem r jako

$$\mathcal{B}(\mathbf{v}, r) := \{\mathbf{u} \in \mathbb{F}_q^n \mid d_H(\mathbf{u}, \mathbf{v}) \leq r\}.$$

Počet prvků libovolné koule v $\mathbb{F}_q^n$ o poloměru r značíme $V_q(n,r)$. Jak známo

$$V_q(n,r) = \sum_{i=0}^{r} \binom{n}{i} (q-1)^i.$$

Počet prvků vychází z následujícího. Tento počet nezávisí na volbě slova z kódu a tedy můžeme uvažovat, že měříme kouli okolo nulového slova. Sčítá se pro 0 změn až po r změn. V i-tém kroku tedy hledáme slova, která se liší v i pozicích. Protože uvažujeme nulové slovo, tak hledáme slova, která mají i nenulových pozic. Celkově je n pozic a tedy vybíráme i pozic z n pozic. Vybrané položky potom mohou nabývat $q-1$ nenulových hodnot a to právě na i pozicích.

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

Důkaz. Na úvod si přepíšeme nerovnost $\frac{r}{n} \leq 1 - q^{-1}$.

$$\begin{aligned}
\frac{r}{n} &\leq 1 - q^{-1} \\
\frac{r}{n} - 1 &\leq -q^{-1} \\
\frac{r-n}{n} &\leq -q^{-1} \\
\frac{n}{r-n} &\geq -q \\
\frac{n}{r-n} + 1 &\geq -q + 1 \\
\frac{n}{n-r} - 1 &\leq q - 1 \\
\frac{n-n+r}{n-r} &\leq q - 1 \\
\frac{r}{(n-r)(q-1)} &\leq 1
\end{aligned}$$

Pro $r = 0$ lemma zřejmě platí. Dostáváme totiž $H_q(\frac{0}{n}) = H_q(0) = 0$ a $V_q(n,0) = 1$, protože slovo, které se liší v 0 položkách je pouze jedno a tedy $\log_q(1) = 0$.

Předpokládejme tedy $r > 0$. Výraz který chceme dokázat si přepišme pomocí exponenciální funkce o základu q. Nerovnost se nezmění, protože exponenciála je rostoucí funkce. Tedy budeme chtít dokázat

$$q^{nH_q(\frac{r}{n})} \geq V_q(n,r).$$

Platí

$$\begin{aligned}
q^{nH_q(\frac{r}{n})} &= q^{n(-\frac{r}{n}\log_q \frac{r}{n} - (1-\frac{r}{n})\log_q(1-\frac{r}{n}) + \frac{r}{n}\log_q(q-1))} \\
&= q^{-r\log_q \frac{r}{n} - (n-r)\log_q(1-\frac{r}{n}) + r\log_q(q-1)} \\
&= \left(\frac{r}{n}\right)^{-r} \left(1 - \frac{r}{n}\right)^{-(n-r)} (q-1)^r \\
&= \frac{n^r}{r^r} \frac{(n-r)^{r-n}}{n^{r-n}} (q-1)^r \\
&= \frac{n^n}{r^r (n-r)^{n-r}} (q-1)^r
\end{aligned}$$

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

$$n^n = (r + (n-r))^n \tag{1}$$

$$= \sum_{i=0}^{n} \binom{n}{i} r^i (n-r)^{n-i} \frac{(q-1)^i}{(q-1)^i} \tag{2}$$

$$\geq \sum_{i=0}^{r} \binom{n}{i} \left(\frac{r}{(n-r)(q-1)} \right)^i (n-r)^n (q-1)^i \tag{3}$$

$$\geq \sum_{i=0}^{r} \binom{n}{i} \left(\frac{r}{(n-r)(q-1)} \right)^r (n-r)^n (q-1)^i \tag{4}$$

První rovnost je pouze vhodně zapsáno n. Druhá rovnost vychází ze vzorce pro binomický rozvoj. Navíc jsme ještě výraz vynásobili zprava vhodně zapsanou jednotkou.

V nerovnosti (3) máme nerovnost díky tomu, že nesčítáme po n ale pouze po r. Platí, že $r < n$ a to díky nerovnosti z předpokladu lemmatu. Díky tomu také víme, že všechny sčítance jsou nezáporné a tedy jsme sumu skutečně zmenšili.

V nerovnosti (4) jsme využili nerovnost, kterou jsme si odvodili na začátku důkazu a sice že $\frac{r}{(n-r)(q-1)} \leq 1$. Nyní dosadíme spodní odhad za n^n a získáme tak

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

Celkově tak máme odhad

$$q^{nH_q(r/n)} \geq \sum_{i=0}^{r} \binom{n}{i} (q-1)^i = V_q(n,r),$$

protože suma je přesně objem koule o poloměru r. □

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. *Pro každý $[n,k]_q$ kód s pokrývacím poloměrem r_c platí*

$$r_c \geq n H_q^{-1}(\alpha / \log_2 q),$$

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

Prvek α je stále relativní kapacita. Máme zde totiž zlomek $\frac{n-k}{n}$, kde $n-k$ je délka zprávy, kterou vkládáme do jednoho bloku a n je relativní počet bitů, které vkládáme do jednoho bloku, neboli počet bitů, které vkládáme do jednoho prvku toho bloku. Logaritmus $\log_2 q$ potom vyjadřuje počet bitů, které vkládáme do jednoho bloku.

Důkaz. Jestliže $\frac{r_c}{n} \geq 1 - q^{-1}$, pak tvrzení věty platí triviálně a to díky tomu, že H_q^{-1} je zobrazení s nejvyšší možnou výstupní hodnotou, která je rovna právě $1 - q^{-1}$.

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

$$\bigcup_{\mathbf{c} \in \mathcal{C}} \mathcal{B}(\mathbf{c}, r_c) = \mathbb{F}_q^n .$$

Toto plyne z představy kterou jsme si již zmiňovali. Nafukujeme kolem každého kódového slova kouli a jakmile bude sjednocení těchto koulí obsahovat celý prostor F_q^n, tak tento poloměr je právě r_c - proto pokrývací poloměr. To znamená, že jinými slovi pro každý $[n,k]_q$ kód s pokrývacím poloměrem r_c platí

$$q^n \leq q^k V_q(n, r_c)$$

a to díky tomu, že s použitím poloměru r_c víme, že když sjednotíme koule kolem všech kódových slov, tak dostaneme $\mathbb{F}_q^n$. Počet prvků v této kouli je roven $V_q(n, r_c)$ a není závislý na slovu okolo kterého je. Počet kódových slov v $\mathcal{C}$ je roven $|\mathcal{C}| = q^k$ a tedy když vynásobíme objem každé koule počtem prvků v kódu $\mathcal{C}$ tak dostaneme alespoň velikost $\mathbb{F}_q^n$, což je q^n (alespoň kvůli tomu, že některé koule se mohou protínat a tedy některé prvky započítáme dvakrát ale klidně i vícekrát). Tento odhad můžeme dále upravit následovně

$$\begin{aligned} q^n &\leq q^k V_q(n, r_c) \\ q^{n-k} &\leq V_q(n, r_c) \\ n - k &\leq \log_q V_q(n, r_c) \end{aligned}$$

Tento logaritmus jsme již odhadli v předchozím lemmatu a máme tedy

$$n - k \leq \log_q V_q(n, r_c) \leq n H_q\left(\frac{r_c}{n}\right)$$

a z toho

$$\begin{aligned} n - k &\leq n H_q\left(\frac{r_c}{n}\right) \\ \frac{n-k}{n} &\leq H_q\left(\frac{r_c}{n}\right) \end{aligned}$$

a protože předpokládáme, že $\frac{r_c}{n} < 1 - q^{-1}$, můžeme aplikovat funkci H_q^{-1} a dostáváme tak celkově

$$H_q^{-1}\left(1 - \frac{k}{n}\right) \leq \frac{r_c}{n}$$

a tedy

$$r_c \geq n H_q^{-1}\left(\frac{\left(1 - \frac{k}{n}\right)\log_2 q}{\log_2 q}\right)$$

jak jsme chtěli dokázat. □

11. Odvoďte horní mez na efektivitu maticového vkládání pro kód s parametry $[n,k]_q$ a její souvislosti s perfektními kódy.
Textový zdroj: skripta - kapitola 2.5 (Tvrzení 2.10, Důsledek 2.11)

Připomeňme, že perfektní kód. Perfektní kód má minimální váhu d a platí, že $r = (d-1)/2$ a $\mathbb{F}_q^n$ je disjunktním sjednocením $\mathcal{B}_q(\mathbf{c}, r)$ přes všechna $\mathbf{c} \in \mathcal{C}$.

Mezi perfektní kódy patří všechny úplné kódy (mají dimenzi $n = k \implies n-k = 0$ a tedy pomocí nich můžeme vkládat jen zprávy délky nula - jsou nepoužitelné), jednobodové kódy a binární kódy liché délky. Tyto jsou známy jako triviální perfektní kódy. Každý netriviální perfektní kód má parametry Hammingova kódu, Golyova $[23, 12, 7]_2$ kódu, anebo Golyaova $[11, 6, 5]_3$ kódu.

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

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

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

Důkaz. Označme $\mathcal{D}$ multimnožinu $\{w(\mathbf{c}-\mathbf{u}) \mid \mathbf{u} \in \mathbb{F}_q^n, \mathbf{c} \in \mathcal{C})\}$. Každá hodnota $i \in \{0, 1, \ldots, n\}$ má v $\mathcal{D}$ četnost $q^k(q-1)^i\binom{n}{i}$, což plyne z následující úvahy.

V multimnožině se díváme na váhy přes všechna kódová slova a přes všechny prvky $\mathbb{F}_q^n$, což je vektorový prostor. Z toho plyne, že takto pro každé $\mathbf{c}$ bude rozdíl $\mathbf{c}-\mathbf{u}$ nabývat všech hodnot $\mathbb{F}_q^n$ a na volbě $\mathbf{c}$ tedy nezáleží. Zvolme si tedy $\mathbf{c} = \mathbf{0}$. Kolik je slov ve vzdálenosti i od $\mathbf{0}$? Jsou to slova, která mají i pozic nenulových (tyto pozice můžeme vybrat $\binom{n}{i}$ způsoby) a nenulových hodnot máme $q-1$ a těchto hodnot nabývá i pozic a tedy $(q-1)^i$. Celkově tak máme pro jedno kódové slovnost četnost váhy i rovnou $(q-1)^i\binom{n}{i}$. Platí $|\mathcal{C}| = q^k$ a tedy četnost váhy hodnoty i v multimnožině je opravdu rovna $q^k(q-1)^i\binom{n}{i}$.

Pro r_a z definice platí, že $r_a = q^{-n}\sum_{\mathbf{u}\in\mathbb{F}_q^n} d_H(\mathbf{u}, \mathcal{C})$, což je rovno $q^{-n}\sum_{\mathbf{u}\in\mathbb{F}_q^n} \min_{\mathbf{c}\in\mathcal{C}}(\mathbf{c}-\mathbf{u})$. Našim cílem proto bude získat spodní odhad výrazu $\sum_{\mathbf{u}\in\mathbb{F}_q^n} \min_{\mathbf{c}\in\mathcal{C}}(\mathbf{c}-\mathbf{u})$. Tento výraz vyjadřuje součet určité q^n-prvkové podmultimnožiny multimnožiny $\mathcal{D}$ (pro každý prvek $\mathbf{u} \in \mathbb{F}_q^n$ se najde nejbližší kódové slovo $\mathbf{c}$). V tomto součtu tedy sčítáme q^n členů z multimnožiny $\mathcal{D}$ a my bychom chtěli udělat spodní odhad tohoto součtu. Tím bude jistě součet q^n nejmenších prvků z $\mathcal{D}$. Podle předpokladu platí

$$V_q(n,r) = \sum_{i=0}^{r} \binom{n}{i}(q-1)^i \leq q^{n-k}$$

$$\sum_{i=0}^{r} q^k(q-1)^i\binom{n}{i} \leq q^n,$$

což znamená, že četnost pro váhy v rozmezí 0 až r nepřesáhne q^n. Každou četnost pro i přenásobíme i (tedy četnost výskytu váhy i přenásobíme vahou i) a tím získáme

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

Nyní stačí obě strany vynásobit q^{-n} a získáme tak námi dokazovaný výraz

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

Nyní je potřeba dokázat souvislost rovnosti s perfektními kódy. Předpokládejme nejprve, že $\mathcal{C}$ je perfektní kód s minimální vahou d. Potom víme, že $r = (d-1)/2$ a $\mathbb{F}_q^n$ je disjunktním sjednocením $\mathcal{B}_q(\mathbf{c}, r)$ přes všechna $\mathbf{c} \in \mathcal{C}$.

Nafoukneme tedy kouli okolo každého kódového slova tak, aby měla každá poloměr $r = (d-1)/2$. Tyto koule jsou disjunktní a jejich sjednocení dá celý prostor $\mathbb{F}_q^n$. To znamená, že každý prvek $\mathbf{u} \in \mathbb{F}_q^n$ leží v některé kouli od kódového slova $\mathbf{c}$. To taky znamená, že $\min_{\mathbf{c}'\in\mathcal{C}} w(\mathbf{c}'-\mathbf{u}) = w(\mathbf{c}-\mathbf{u})$. Tedy pro každou kouli můžeme spočítat četnosti od každé vzdálenosti z $0, \ldots, r$ a vynásobit tuto četnost příslušnou vahou=vzdáleností.

Ještě jinak, uvažujme $\mathcal{B}_{\mathbf{c}}$ kouli okolo kódového slova $\mathbf{c} \in \mathcal{C}$. Potom pro každý prvek z této koule platí

$$\sum_{\mathbf{u}\in\mathcal{B}_{\mathbf{c}}} \min_{\mathbf{c}'\in\mathcal{C}} w(\mathbf{c}'-\mathbf{u}) = \sum_{\mathbf{u}\in\mathcal{B}_{\mathbf{c}}} w(\mathbf{c}-\mathbf{u}) = \sum_{i=0}^{r} i(q-1)^i \binom{n}{i}.$$

Kódových slov je q^k, což je i zároveň počet disjunktních koulí a tedy máme v dokazovaném výrazu rovnost.

Nyní předpokládejme, že platí rovnost. Potom vlevo sčítáme právě q^n nejmenších prvků z $\mathcal{D}$ a to je rovno pravé straně, kde sčítáme postupně vzdálenosti $0,\dots,r$. Zároveň pravá strana je největší jak jen to jde a tedy se zde sčítá právě q^n prvků a levá strana je nejmenší jak jen to jde, takže se zde sčítají právě nejmenší prvky z $\mathcal{D}$, což se děje i vpravo. Z toho plyne, že $\min_{\mathbf{c}\in\mathcal{C}} w(\mathbf{c}-\mathbf{u}) \leq r$ a proto každé $\mathbf{u} \in \mathbb{F}_q^n$ leží ve sjednocení $\bigcup_{\mathbf{c}\in\mathcal{C}} \mathcal{B}_q(\mathbf{c}, r)$.

Sčítáme q^n prvků napravo i nalevo. Napravo jsme počet sčítanců odhadovali pomocí $q^k V_q(n,r) \leq q^n$, musí tedy platit rovnost a tedy $q^k V_q(n,r) = q^n$. To znamená, že součet objemů q^k koulí nám dá celý prostor $\mathbb{F}_q^n$ a tedy jsou koule disjunktní a tedy máme perfektní kód. □

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

Důkaz. Již víme, že $e = (n-k)(\log_2 q)/r_a$. Dosadíme tedy do tohoto vztahu náš odhad pro r_a

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

a získáme tak

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

□

12. Vysvětlete, co to jsou SDCS a jakým způsobem se používají ve steganografii. Odvoďte spodní mez na maximální počet změn vyvolaných SDCS vkládáním.
Textový zdroj: skripta - kapitola 3 (v podstatě celá, Věta 3.2), prezentace - „Součtově a rozdílově pokrývací množiny“

Zkratka SDCS vychází z anglického „sum and difference covering set“, což se předkládá do češtiny jako „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 bloky (x_1, x_2) a do každého bloku vkládá kvaternární symbol $z \in \mathbb{Z}_4$.

$x_1 + 2x_2$	0	1	2	3
0	$(0,0)$	$(+1,0)$	$(0,\pm1)$	$(-1,0)$
1	$(-1,0)$	$(0,0)$	$(+1,0)$	$(0,\pm1)$
2	$(0,\pm1)$	$(-1,0)$	$(0,0)$	$(+1,0)$
3	$(+1,0)$	$(0,\pm1)$	$(-1,0)$	$(0,0)$

(čísla nad sloupci značí vkládanou zprávu z). Extrakce je definována jako $\mathrm{Ext}(y_1, y_2) = (y_1 + 2y_2) \bmod 4$. Vkládání probíhá tak, že nejdříve provedeme extrakce na bloku nosiče a ověříme, zda je výsledek roven z. Jestliže není, změníme blok nosiče tak, jak je uvedeno v tabulce.

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 $e = 2/\frac{3}{4} = \frac{8}{3} \approx 2,667$ 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

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

pak říkáme, že $\mathcal{A}$ je *součtově a rozdílově pokrývací množina* s parametry (n, r, G).

První vztah nám v závislosti na parametru r říká, kolik prvků z množiny $\mathcal{A}$ můžeme maximálně použít pro získání všech prvků z G. Druhý vztah nám potom říká, že z množiny $\mathcal{A}$ musíme být schopni nagenerovat libovolný prvek z G.

Definice. 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 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 $z \in G$

`VYSTUP:`

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

`ALGORITMUS:`

(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.

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

□

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

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

za předpokladu, že zprávy jsou voleny z G náhodně s rovnoměrným rozdělením.

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

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

Nerovnost proto, protože jeden prvek bychom mohli získat různými kombinacemi prvků z $\mathcal{A}$. Z tohoto můžeme odvodit obdobu věty z otázky 10.

Věta. *Pro každou (n, r, G)-SDCS platí*

$$r \geq n H_3^{-1}(\alpha / \log_2 3).$$

Důkaz. Důkaz provedeme obdobně. Víme, že $H_q^{-1} : [0, 1] \to [0, 1 - q^{-1}] = [0, \frac{2}{3}]$ a tedy pro $\frac{r}{n} \geq \frac{2}{3}$ platí tvrzení věty triviálně. Dále tedy předpokládejme $\frac{r}{n} < \frac{2}{3}$. Podívejme se na nerovnost, kterou jsme měli před touto větou

$$|G| \leq V_3(n, r).$$

Toto můžeme zlogaritmovat

$$\log_3 |G| \leq \log_3 V_3(n, r).$$

Z lemmatu v otázce 10 víme, že $\log_q V_q(n, r) \leq n H_q(\frac{r}{n})$ a tedy v kombinací s nerovností dostaneme

$$\log_3 |G| \leq \log_3 V_3(n, r) \leq n H_3\left(\frac{r}{n}\right).$$

Protože se pohybujeme na intervalu $[0, \frac{2}{3}]$, tak víme, že H_3 je rostoucí a tedy můžeme aplikovat inverz (a ještě nerovnost vydělíme n).

$$\frac{r}{n} \geq H_3^{-1}\left(\frac{\log_3 |G|}{n}\right) \implies r \geq n H_3^{-1}\left(\frac{\log_3 |G|}{n}\right)$$

Nakonec dosadíme

$$\log_3 |G| = \frac{\log_2 |G|}{\log_2 3} = \frac{n\alpha}{\log_2 3}$$

a tím získáme výslednou nerovnost

$$r \geq nH_3^{-1}(\alpha/\log_2 3).$$

□

13. Formulujte a dokažte větu o mokrém nosiči.
Textový zdroj: skripta - kapitola 4 (Věta 4.1), prezentace - „Psaní na mokrý papír“

Nejprve se podívejme na motivaci. Už jsme se setkali s problémem, kdy je nežádoucí vkládat do některých prvků nosiče (PNG/GIf: oblasti s nízkou texturou, JPEG: nulové AC koeficienty). Ukázali jsme si řešení, kdy jsem problematické prvky přeskakovali. Co se nám ale dělo např. u JPEGu bylo, že nám tam vznikaly falešné nulové AC koeficienty, které se nečtou a tedy jsme museli vkládat daný bit znovu. Toto jsme nazývali smršťování a docházelo tak k omezení kapacity nosiče. Nyní si ukážeme způsob, jak se přiblížit libovolně blízko k tomu, abychom využili celou délku použitelné části z nosiče (tedy mohli vložit zprávu dlouhou n - „počet všech problematických pixelů v nosiči“).

Mokrými pixely budeme nazývat mokrými - ty nesmíme upravovat a některé suchými - ty smíme upravovat. Názvosloví pochází (jak již název napovídá) od psaní na částečně mokrý papír. Tam kde je mokrý, tak tam nepíšeme, protože by došlo k rozpití.

Tedy naše situace je následující. Máme obrázek. Některé pixely jsou mokré a některé suché. Upravíme obrázek (ukážeme si jak) a odešleme ho. Ten ale cestou zaschne. Příjemce nepozná, kam jsme vložili zprávu, protože neví, které pixely byly suché a které nikoliv. Ale i tento problém se dá vyřešit.

Mějme následující větu. Ta nám říká, že můžeme mít libovolný poměr suchých prvků σ v nosiči a libovolné $\epsilon > 0$ a $\delta > 0$. Potom existuje délka nosiče n taková, že budeme moct vložit zprávu délky $(\sigma - \epsilon)n$ (tedy využijeme téměř všechny suché prvky) a tuto informaci budeme do takového nosiče moct vložit s pravděpodobností $1 - \delta$ (což se blíží jedné).

Věta. *Pro každé $\epsilon > 0, \delta > 0$ 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 $1-\delta$ vložit informaci velikosti $(\sigma-\epsilon)n$ symbolů.*

Důkaz. Na začátku důkazu si zavedeme některé struktury. Nechť Σ je množina symbolů a $q := |\Sigma|$. (Typicky $\Sigma = \mathbb{F}_2$ nebo $\Sigma = \mathbb{F}_3$.) Nechť Z je množina zpráv. Tato množina má velikost $|Z| = q^{(\sigma-\epsilon)n}$. Velikost máme z toho, že chceme vložit zprávu délky $(\sigma - \epsilon)n$ a máme q různých symbolů.

Nyní zkonstruujeme stegosystém. Nejprve vytvoříme kódovou knihu $\mathcal{B} = \{\mathcal{P}_z : z \in Z\}$. Zatím v knize nic není. Jediné co zatím máme je, že kniha má $|Z|$ stránek a každá stránka je indexována podle nějaké zprávy z.

Teď naplníme tuto knihu. Rozmístíme proto zcela náhodně slova z množiny Σ^n, tak aby na každé stránce bylo $q^n/q^{(\sigma-\epsilon)n} = q^{(1-\sigma+\epsilon)n}$ slov (q^n je počet všech možných slov a $q^{(\sigma-\epsilon)n}$ je počet stránek). Každé slovo se nachází pouze na jedná stránce a tedy platí $\Sigma^n = \bigcup_{z\in Z} \mathcal{P}_z$, kde toto sjednocení je disjunktní. Důležité je, že odesílatel i přijímač si tuto kódovou knihu dopředu nasdílejí.

Jaká je tedy naše situace. Máme knihu, kde každá stránka je indexována podle zprávy z a každé stránce se nachází $q^{(1-\sigma+\epsilon)n}$ náhodných slov délky n. Tohoto budeme chtít využít k následujícímu. Uvažujme, že máme zprávu $z \in Z$ a chceme ji odeslat, ale nechceme měnit mokré indexy. Podíváme se tedy na stránku $\mathcal{P}_z$ a tam najdeme slovo takové, které se bude shodovat na indexech mokrých pixelů s mokrými pixely v nosiči. Toto slovo vezmeme, vložíme ho do nosiče (tím nezměníme mokré pixely) a odešleme stegoobrázek. Příjemce se podívá do stegoobrázku, z něj zjistí vložené slovo, podívá se do knihy, najde stránku s tímto slovem a index této stránky je ona tajná zpráva, kterou chtěl odesílatel sdělit. Jaká je ale pravděpodobnost, že takové vhodné slovo právě na stránce $\mathcal{P}_z$ najdeme? Chtěli bychom ukázat, že tato pravděpodobnost je alespoň $1 - \delta$ (podle znění věty).

Ukažme si ještě jednou (trochu přehledněji) vkládání a extrakci. Vkládání:

- Vstup: nosič $x \in \Sigma^n$ s $n - \sigma n$ mokrými prvky a zpráva $z \in Z$.
- Pokusíme se na stránce $\mathcal{P}_z$ najít n-tici, která se na mokrých pozicích shoduje s nosičem.
- Podaří-li se, je nalezená n-tice stegoobjektem a mokré pozice zůstávají nezměněny.

Extrakce:

- Vstup: Stegoobjekt $y \in \Sigma^n$.
- Vyhledáme y v knize. Podíváme se na index stránky, ne které jsme y našli a zpráva = index.

Podaří se tedy vkládacímu algoritmu najít na stránce $\mathcal{P}_z$ n-tici, která má na $n - \sigma n$ určených pozicích určené hodnoty? V celé knize je takových n-tic $q^{\sigma n}$ (mokré pozice jsou fixovány a suchých pozic je právě σn). Jaká je pravděpodobnost, že žádná z těchto n-tic není v $\mathcal{P}_z$? Víme, že $\mathcal{P}_z$ vznikla

náhodným výběrem $q^{n-\sigma n+\epsilon n}$ prvků z Σ^n. Pravděpodobnost selhání δ je méně než

$$\delta < \left(\frac{q^n - q^{\sigma n}}{q^n}\right)^{q^{n-\sigma n+\epsilon n}}.$$

Tento vztah plyne z následujícího. Uvažujme, že někdo má v koši $|\Sigma|^n = q^n$ prvků. Zároveň uvažujme (trochu zvláštně), že vidíme do budoucnosti a tedy „víme“ na jakých pozicích bychom potřebovali mít konkrétní hodnoty. Ten někdo náhodně bere slova z koše a lepí je stránky knihy $\mathcal{B}$ a my si jen říkáme„to zase vybral špatně, a zase,...“. Pravděpodobnost, že tento člověk hned první zprávu vylosuje špatně je rovna

$$\frac{q^n - q^{\sigma n}}{q^n},$$

protože ve jmenovateli je počet špatných slov („celkový počet slov“ - „počet dobrých slov“) a v čitateli celkový počet slov. Když ten někdo bude losovat podruhé, tak pravděpodobnost, že vylosuje něco blbého klesne (v prvním losování zmenšil množinu špatných slov v koši). My ale na toto klesání nebudeme brát zřetel a budeme uvažovat, že v každém losování vylosoval něco špatného s pravděpodobností jako při prvním tahu. Díky tomu máme mezi δ a výrazem napravo ostrou nerovnost. Nám ale tento hrubý odhad nebude vadit. Výraz napravo si můžeme přepsat následovně

$$\delta < \left(\frac{q^n - q^{\sigma n}}{q^n}\right)^{q^{n-\sigma n+\epsilon n}} = \left(\left(1 + \frac{-1}{q^{n(1-\sigma)}}\right)^{q^{n(1-\sigma)}}\right)^{q^{\epsilon n}}$$

Tímto jsme na pravé straně získali uvnitř výraz

$$\left(1 - \frac{1}{q^n}\right)^{q^n},$$

(kde $q > 1$), což jde pro $n \to \infty$ k e^{-1}. Tedy od jistého n je výraz

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

menší než (např.) $\frac{1}{2}$ a tedy od jistého n platí

$$\delta < \left(\frac{1}{2}\right) q^{\epsilon n} \to 0$$

a tedy pravděpodobnost úspěchu $1 - \delta$ jde pro $n \to \infty$ k 1. □

14. Vysvětlete, jakým způsobem se ve steganografii využívají konvoluční kódy a Viterbiho algoritmus.
Textový zdroj: skripta - kapitola 5 (Algoritmus 5.3), prezentace - „Vkládání pomocí Viterbiho kódů“

Cílem je využít teorii konvolučních kódů. Prvku nosiče budeme přiřazovat váhu, která bude udávat jeho citlivost na změnu. Nebudeme nutně minimalizovat počet změn v nosiči, ale celkovou váhu změn. Viterbiho dekodér je soft-descision dekodér (rozhoduje se na základě nějakých pravděpodobností).

Značení. Nosič rozdělujeme na bloky n hodnot z konečného tělesa $\mathbb{F}_q$. Rozdíl oproti maticovému vkládání je, že

- Nepracujeme s jednotlivými bloky samostatně.
- Sestrojíme z nich posloupnost vektorů $\{\mathbf{x}_i\}_{i=0}^{\ell-1}$, kde $\mathbf{x}_i = (x_i^{(1)}, \ldots, x_i^{(n)})^T \in \mathbb{F}_q^n$.

Máme tedy nosič $\{\mathbf{x}_i\}_{i=0}^{\ell-1}$ a stegoobjekt $\{\mathbf{y}_i\}_{i=0}^{\ell-1}$. Zpráva je obecně posloupnost vektorů $\{\mathbf{z}_i\}_{i=0}^{\ell-1}$, kde $\mathbf{z}_i \in \mathbb{F}_q^m$.

- My se omezíme na případ $m = 1$.
- Zpráva je pak posloupnost jednosložkových vektorů.
- Budeme ji psát jako posloupnost skalárů $\{\mathbf{z}_i\}_{i=0}^{\ell-1}$.

Proč se omezujeme na $m = 1$? Složitost Viterbiho algoritmu neúměrně narůstá s m. Relativní kapacita nosiče je $\alpha = \frac{m \log_2 q}{n}$, čili jsme omezeni na hodnoty tvaru $\alpha = \frac{\log_2 q}{n}$. Nevadí. Ve steganografii obvykle cílíme na α blízké 0, ale nicméně, zobecnění pro $m > 1$ je jednoduché.

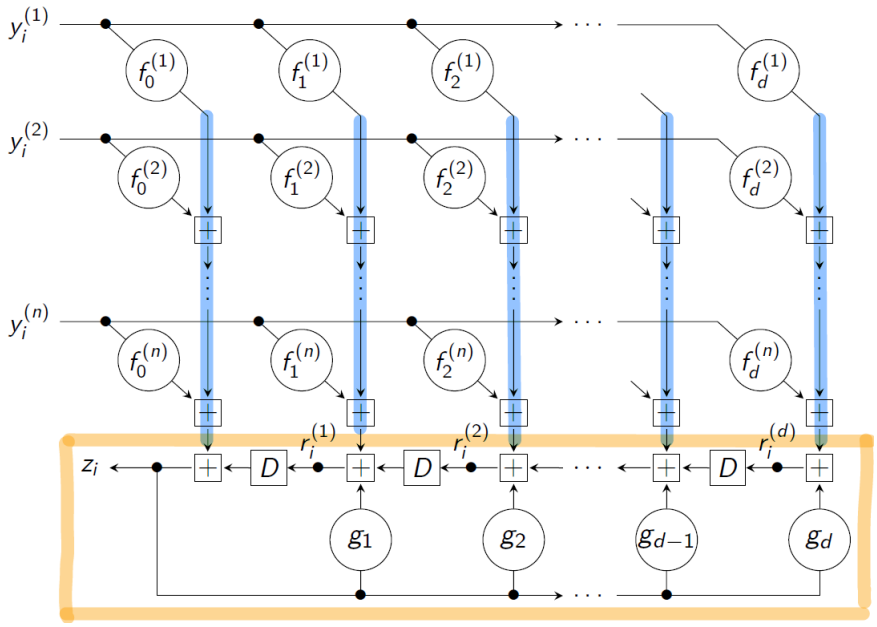
Konvoluční extrakce. Extrakce zprávy se provádí konvolučním překladačem. Implementace je pomocí konvolučního dekodéru

- Kontrolorova normální forma: n paměťových registrů.
- Pozorovatelova normální forma: $m = 1$ paměťových registrů.

(něco jako posuvné registry z kryptografie, uvidíme později).

- $r_i^{(j)}$ je hodnota j-té buňky registru na konci i-tého kroku
- d je délka registru
- D je operátor zpoždění (neřešme co to je)
- Definujme $r_0^{(j)} = 0$ pro $1 \leq j \leq d$ (počáteční stav).
- Hodnoty $f_j^{(k)} \in \mathbb{F}_q$ a $g_j \in \mathbb{F}_q$ jsou konstanty a $g_0 = 0$.

Na tomto schématu vidíme obecně zakreslený konvoluční překladač v pozorovatelově normální formě. Popišme si tento nákres.



Kroužky značí násobení příslušným prvkem nad $\mathbb{F}_q$ a + ve čtverci je sčítání nad $\mathbb{F}_q$. Celý tento překladač je dán polynomy: $f(D) = (f^{(1)}, \ldots, f^{(n)})$ a polynomem $g(D)$. Tyto dva prvky nám tak dávají celkovou stavbu překladače. Do něj potom vstupují vektory $\mathbf{y}_i$ jež je tvořen n bloky a z těchto n bloků získáme překladačem hodnotu z_i. Oranžově zvýrazněná oblast je posuvný registr a bez prvků nad ním se jedná o registr se kterým jsme se již potkali v kryptografii.

Nyní si popíšeme následující Algoritmus.

Algoritmus (konvoluční extrakce)

`VSTUP:`

stegoobjekt $\{\mathbf{y}_i\}_{i=0}^{\ell-1}$ z $\mathbb{F}_q^n$

parametry $f_j^{(k)} \in \mathbb{F}_q$ a $g_j \in \mathbb{F}_q$

`VYSTUP:`

zpráva $\{z_i\}_{i=0}^{\ell-1}$

`ALGORITMUS:`

(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` $\{z_i\}_{i=0}^{\ell-1}$

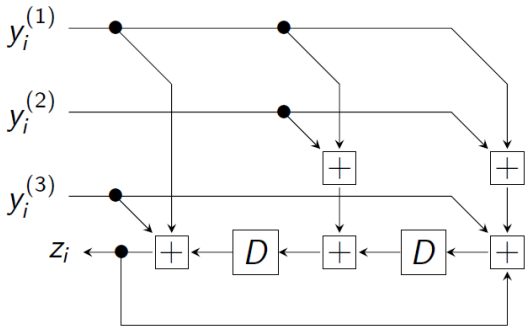
Tento algoritmus vykonává to, co je na schématu výše. V nákresu se nevyskytují s_i. Jedná se o pomocné proměnné, které „probíhají“ každá v samostatném sloupci, který je v nákresu modře zvýrazněn.

Když dorazíme do kroku (5), tak již mám spočítanou celou konstrukci nad oranžovým rámečkem a máme tak v $(s_1, \ldots, s_d)$ hodnoty, které vstupují do posuvného registru. Jak vidíme z nákresu, hodnota s_0 je rovna hodnotě z_i a zároveň hodnota s_0 vstupuje úplně dolů a ovlivní přes $(g_1, \ldots, g_d)$ hodnoty $(s_0, \ldots, s_d)$ (krok (6)) a poté dojde k posunu (krok (7)).

V nákresu jsou taky hodnoty $r_i^{(j)}$, které se v popisu algoritmu nevyskytují. Nahradili jsme je více méně $(s_0, \ldots, s_d)$. V kroku (1) do nich dosadíme nuly a v kroku (7) si do nich uložíme hodnoty pro další běh for cyklu od i.

Máme tedy extrakční algoritmus, který umíme popsat jako konečný překladač. Jak ale vyřešit vkládání? Pro tento proces jsme si zavedeme některé pojmy. Budeme potřebovat trelážový graf. Obecně cesty v tomto grafu symbolizují všechny možné vstupy překladače. Našim trikem bude, že graf sestavíme tak, aby obsahoval pouze ty cesty (vstupy), jejichž výstupem je zpráva, kterou chceme vložit. Tyto cesty jsou kandidátky na stegoobjekt. Viterbiho algoritmus potom najde vestu, která se nejméně liší od nosiče, respektive má nejmenší váhu (distorzi).

Celý proces vkládání si odvodíme z následujícího příkladu. Začněme nejprve průběhem extrakce. Předpokládejme, že máme extrakční algoritmus nad $\mathbb{F}_2$ s parametry $\mathbf{f}(D) = (1+D+D^2, D+D^2, 1+D^2)$ a $g(D) = D^2$. Z těchto údajů získáme následující schéma.



Například první řádek odpovídá $1 + D + D^2$ a tedy vstupní hodnota ovlivní tři hodnoty. Druhý řádek odpovídá $D + D^2$ a tedy ovlivní pouze 2. a 3. hodnotu.

Uvažujme dále, že se nacházíme v i-tém kroku a tedy máme stavy z $i-1$-ního kroku ve tvaru $r_{i-1}^{(1)} r_{i-1}^{(2)} = 10$. Dále uvažujme vstup $\mathbf{y}_i = (1,0,1)^T$.

Protože $r_{i-1}^{(1)} r_{i-1}^{(2)} = 10$, tak víme, že v $(s_0, \ldots, s_d)$ nám na konci $i-1$-ního kroku zůstaly hodnoty $(1,0,0)$. Proč? První dvě jsou ona r a poledí je 0, která se zde dostala z posunutí registru v $i-1$-ním kroku. Vidíme, že n je v našem případě rovno 3 a spustíme for cyklus v (3).

$$\begin{aligned} j = 1 \qquad & (s_0, s_1, s_2) := (1,0,0) + 1(1,1,1) = (0,1,1) \\ j = 2 \qquad & (s_0, s_1, s_2) := (0,1,1) + 0(0,1,1) = (0,1,1) \\ j = 3 \qquad & (s_0, s_2, s_2) := (0,1,1) + 1(1,0,1) = (1,1,0) \end{aligned}$$

Tedy bit zprávy z_i je roven $s_0 = 1$. Tento bit zároveň vstoupí dolů do posuvného registru a provede se krok (6) a tedy

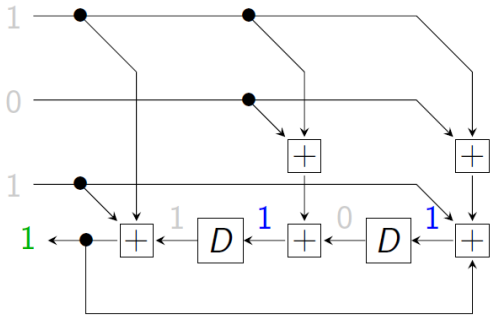
$$(s_0, s_1, s_2) := (1,1,0) + 1(0,0,1) = (1,1,1).$$

Nyní se provede posun v kroku (7)

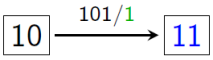
$$(s_0, s_1, s_2) := (1,1,0)$$

a tedy hodnoty $r_i^{(1)} r_i^{(2)} = 11$.

Na nákresu můžeme vidět celou situaci s hodnotami. Šedé jsou vstupní hodnoty. Modré jsou hodnoty jsou stav registru na konci i-tého kroku a zelená hodnota je výstupní bit z_i.

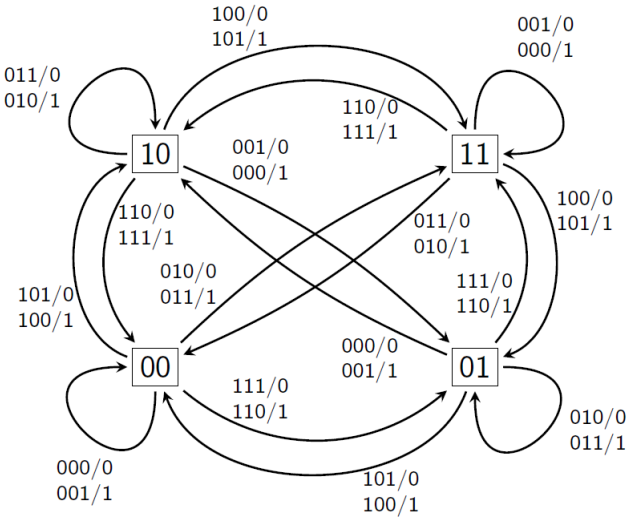


Celou situaci z příkladu potom zapisujeme následovně.

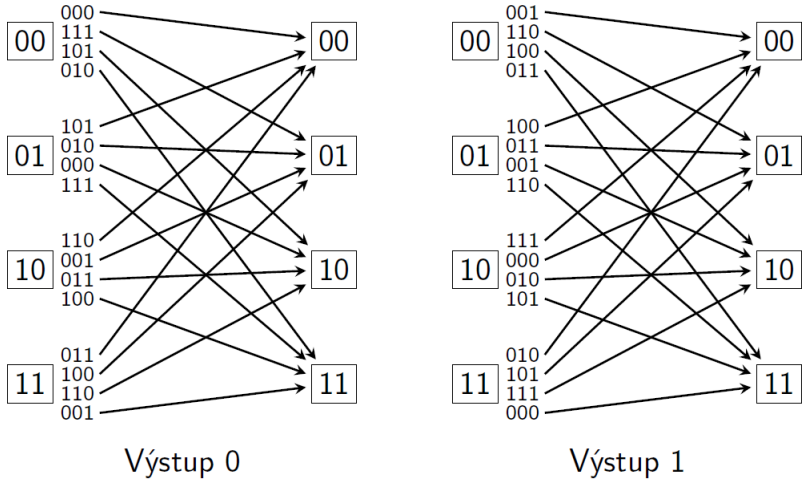


Tedy, že ze stavu 10 jsme se dostali do stavu 11 za použití vstupu 101 a přitom nám vznikl bit zprávy 1.

Nyní bychom mohli zkusit dosadit za počáteční stav všechny kombinace a za vstup do nich také všechny možné trojice. Takto bychom obdrželi následující graf.

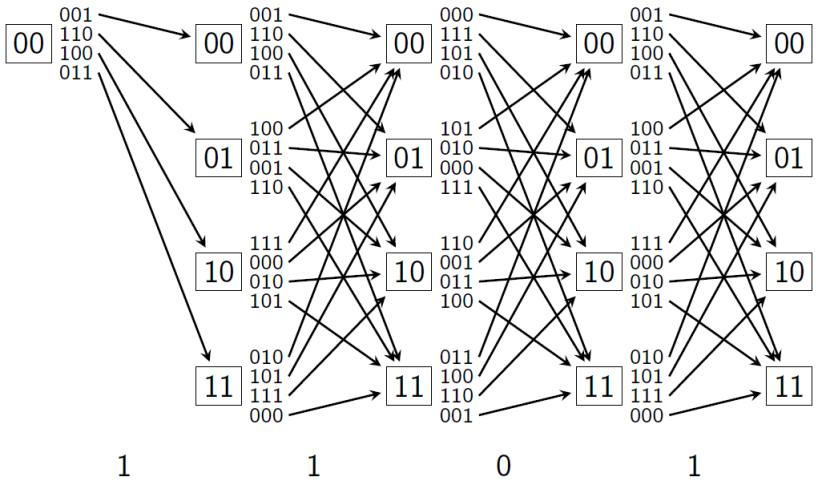


Tyto přechody můžeme rozdělit do dvou trelážových modulů podle jejich výstupní hodnoty (treláž je mimo jiné taková ta dřevěná šikmá mřížka na které rostou popínavé rostliny na zahradě).



Obrázek 18. Trelážové moduly

Předpokládejme nyní, že chceme vložit zprávu 1101. K tomu si sestavíme následující trelážový graf (pořád pracujeme s překladačem z příkladu).



Proč vypadá zrovna takto? V prvním sloupci se nachází pouze stav 00 a to proto, že tento stav je vždy výchozí. Prvním výstupem má být 1. a ze stavu 00 se dá získat výstupní bit 1 pouze, pokud vložíme zprávu 001, 110, 100 nebo 011. A tak podobně dále. V grafu tak máme spoustu cest z nichž ale každá dá na výstupu námi požadovanou zprávu 1101. Jak ale vybrat tu nejlepší? Chceme aby Viterbiho algoritmus našel cestu treláží takovou, která se nejvíce „podobá“ nosiči.

Algoritmus si nejprve popíšeme a poté ukážeme na konkrétním příkladu.

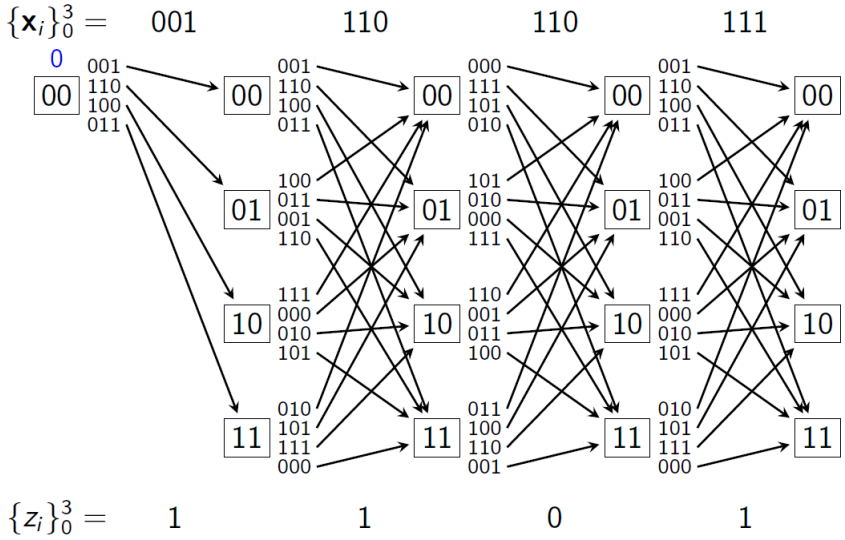
- V první fázi procházíme treláž zleva doprava, tj. pro $i = 0, \ldots, \ell - 1$.
  - Pro každé i máme až q^d možných stavů překladače.
  - Pro každý z těchto stavů spočítáme váhu nejlehčí cesty, která do něho vede a zaznamenáme poslední hranu cesty.
- Jakmile dojdeme na konec treláže, vybereme stav, do kterého vede nejlehčí cesta.
- Zpětným průchodem tuto cestu zrekonstruujeme.

Průběh algoritmu si předvedeme na příkladu nosiče

$$\{\mathbf{x}_i\}_{i=0}^{3} = \{(0,0,1)^T, (1,1,0)^T, (1,1,0)^T, (1,1,1)^T\}$$

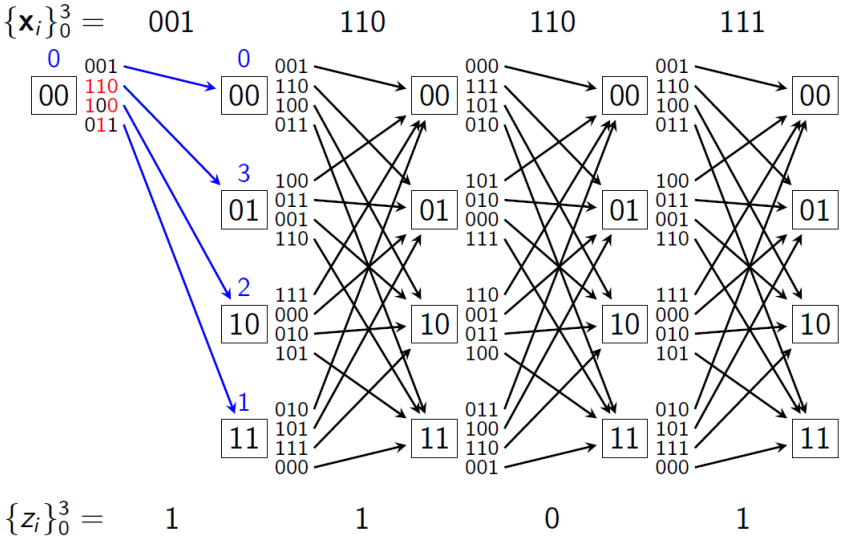
Je potřeba ještě definovat metriku, kterou budeme používat. My v tomto případě použijeme Hammingovu metriku (tedy Hammingovu váhu dvou vektorů).

Pro začátek si situaci zakreslíme do trelážového grafu. Do grafu budeme modře zaznamenávat váhy a to podle toho, s jakou váhou jsme se dostali do daného stavu. Začínáme ve stavu 00. Do něj jsme se tedy dostali bez jakéhokoliv problému.



Nyní si ohodnotíme druhý sloupec. Jak na to? V prvním sloupci máme pouze jeden stav a z něj vedou právě čtyři cesty. Tyto cesty jsou označeny některými hodnotami $(001, 110, \ldots)$. To jsou hodnoty potencionálního stegoobrjektu. Vzpomeňme si na předkladač do kterého jsme vkládali $\mathbf{y}_i$ a z něj vypadl nějaký bit s končil v nějakém stavu v registru. A přesně to se teď snažíme udělat, ale co nejvhodněji vzhledem k nosiči.

První tři bity nosiče jsou $\mathbf{x}_1 = (0,0,1)$. Tedy pokud bychom použili $\mathbf{y}_1 = (0,0,1)$, tak máme nula rozdílů mezi těmito vektory. Stav do kterého vede příslušná cesta (tedy stav 00 v druhém sloupci) tak ohodnotíme 0+0 (první nula je za váhu, kterou jsme se dostali do stavu v prvním sloupci a druhá nula je za počet změn oproti $\mathbf{x}_1$). Dalším je 110 a ten se od $\mathbf{x}_1$ liší ve všech třech pozicích a tedy stavu v druhém sloupci do kterého vede tato cesta přiřadíme váhu $0+3$. Pro zbylé dvě cesty obdobně. Tyto cesty si zapamatujeme. Získáme tak následující ohodnocení.



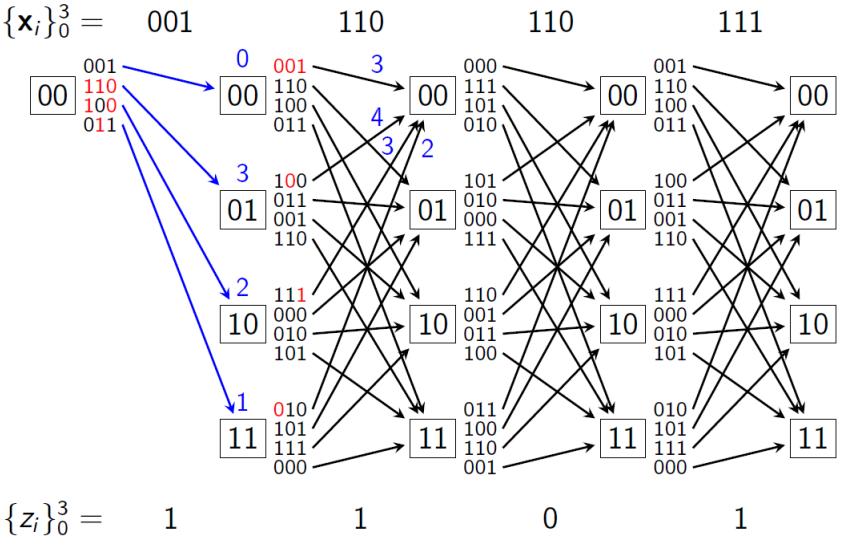
Ono správným principem je podívat se na stavy ve sloupci, brát z něj postupně jednotlivé stavy a pro tento stav se podívat na všechny cesty, které do něj vedou a vybrat tu nejlepší. U třetího sloupce už se to dá ale názorně ukázat.

Vezmeme si tedy stav 00 v třetím sloupci. Do něj vedou čtyři cesty (každá z jiného stavu v druhém sloupci). První cesta je ze stavu 00 v druhém sloupci. Tu můžeme použít, pokud bychom zvolili $\mathbf{y}_2 = (0,0,1)$. To by ale znamenalo, že se od $\mathbf{x}_2$ lišíme ve všech třech pozicích a tedy váha cesty z 00 ve druhém sloupci do 00 ve třetím bude $0+3$ (nula za cestu do stavu 00 v druhém sloupci a 3 za použití $(0,0,1)$). Prozatím ještě nepřiřazujeme váhu stavu 00 ve třetím sloupci.

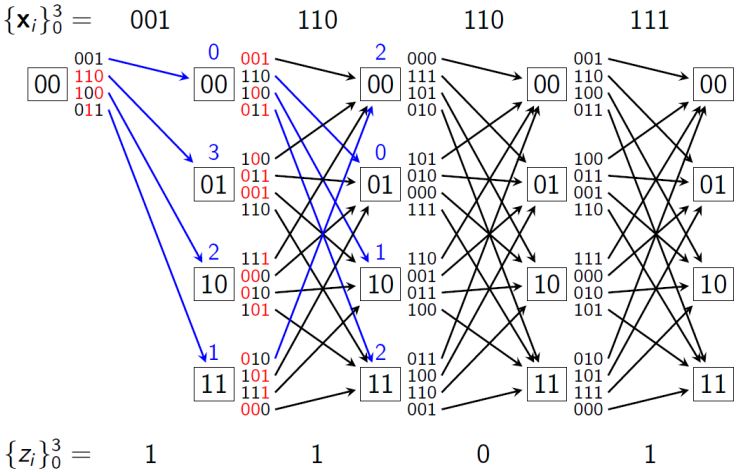
Druhá cesta je ze stavu 01 ve druhém sloupci. Abychom se z tohoto stavu dostali do stavu 00 ve třetím sloupci, tak bychom museli použít $(1,0,0)$, což se liší od $\mathbf{x}_2$ v jedné pozici. Tedy váha této cesty je $3+1$ (3 za cestu do stavu 01 ve druhém sloupci a 1 za použití $(1,0,0)$).

Takto ohodnotíme i zbylé cesty. Cesty tak mají váhy $3,4,3,2$. Nejlepší je tedy cesta ze stavu 11 ve druhém sloupci. Tuto cestu si zapamatujeme a stav 00 ve třetím sloupci ohodnotíme právě vahou 2. Samozřejmě se může někdy stát, že vede do nějakého stavu více cest se stejnou váhou. Potom se můžeme rozhodnout náhodně, protože obě cesty jsou rovnocenně dobré.

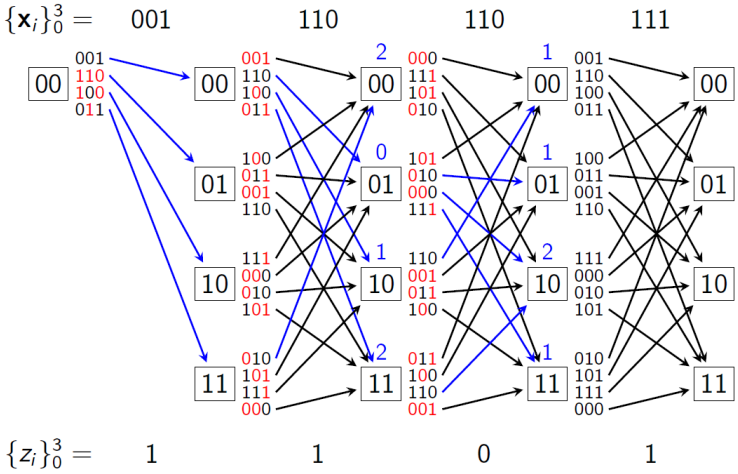
Všimněme si také, že váha 0 u stavu 00 v prvním sloupci zmizela. Není si ji totiž už nutné pamatovat.



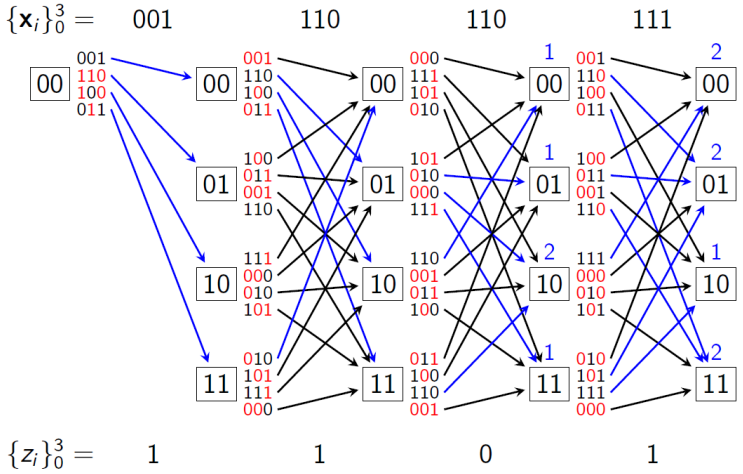
Stejným způsobem ohodnotíme i zbylé stavy v třetím sloupci.



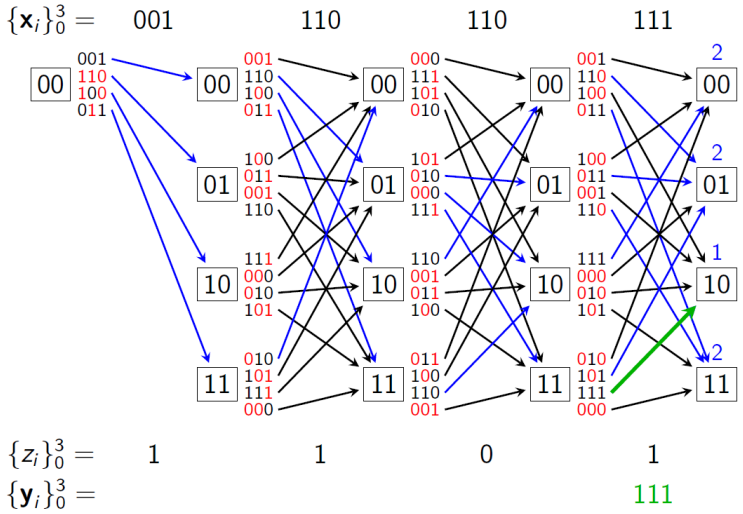
A takto ohodnotíme stavy i ve čtvrtém sloupci.



A taky v posledním sloupci.



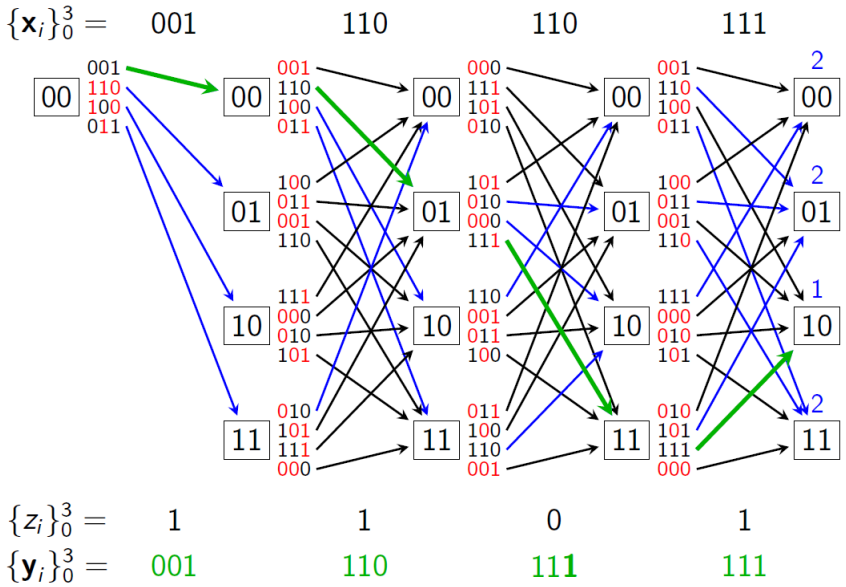
Nyní máme ohodnoceny stavy v posledním sloupci. Vidíme, že tři mají váhu 2 a jeden má váhu 1. Tento stav si zvolíme a započneme zpětnou fázi. Díky tomu, že si pamatujeme ty nejlepší cesty mezi stavy, tak jsme schopni zpětně získat cestu, kterou jsme se dostali do stavu 10 v posledním sloupci. Do tohoto stavu jsme ze čtvrtého sloupce přišli z stavu 11 a to za použití $(1, 1, 1)$.



Takto budeme postupovat až získáme celou cestu až do stavu počátečního stavu 00 v prvním sloupci. K čemu je to ale vlastně dobré? Jak vidíme, použili jsme vektory

$$(0, 0, 1) \quad (1, 1, 0) \quad (1, 1, 1) \quad (1, 1, 1)$$

a vidíme, že pokud tyto vektory použijeme jako stegoobjekt, tak se od nosiče bude lišit pouze v jedné pozici a bude pro něj platit, že extrakční překladač z něj získá námi požadovanou posloupnost bitů $z_1, z_2, z_3, z_4 = (1, 1, 0, 1)$.



Časová složitost algoritmu.

- Procházíme ℓ modulů zleva doprava.
- V každém modulu projdeme (až) q^d stavů a spočítáme váhu nejlehčí cesty, která do akždého stavu vede.
  - Pro každý stav tedy zvážíme všechny vstupující hrany.
  - Do každého stavu vede v průměru $\frac{1}{q} \cdot q^n$ hran.
  - Zvážení jedné hrany vyžaduje n porovnání.
- Zpětný průchod má složitost $\mathcal{O}(\ell)$ operací. (Zanedbatelné.)
- Časová složitost algoritmu je $\mathcal{O}(q^{d+n-1}\ell n)$ skalárních operací.
- Složitost ještě vylepšíme na $\mathcal{O}(q^{d+2}\ell n)$ vektorových operací tak, že rozvineme treláž.

Všimněme si, že v našem příkladu by nedošlo ke zlepšení, protože jsme počítali s $n = 3$. Podívejme se ještě na trelážové moduly. V nich jsme měli přechody mezi stavy za použití nějakých vstupních vektorů. My si nyní tyto přechody ještě podrobněji rozebereme. Tedy, na každý trelážový modul aplikujeme takzvané rozvinutí modulů (tedy si podrobněji rozpitváme jednotlivé cesty mezi stavy). Tedy každý přechod automatu rozvineme na $n+1$ podkroků, čili každý modul rozvineme na $n+1$ podmodulů. K rozvinutí využijeme již uvedeného extrakčního překladače. Toto rozvinutí pak použijeme k formulaci Viterbiho algoritmu.

Připomeňme proto již zmíněné věci. Naše schéma příkladu a výpočty (s_0, s_1, s_2).

Algoritmus (konvoluční extrakce)

VSTUP:

stegoobjekt $\{\mathbf{y}_i\}_{i=0}^{\ell-1}$ z $\mathbb{F}_q^n$

parametry $f_j^{(k)} \in \mathbb{F}_q$ a $g_j \in \mathbb{F}_q$

VYSTUP:

zpráva $\{z_i\}_{i=0}^{\ell-1}$

ALGORITMUS:

(1) $(s_0, \ldots, s_d) := (0, \ldots, 0)$

(2) for $i = 0, \ldots, \ell - 1$ do

(3) for $j = 1, \ldots, n$ do

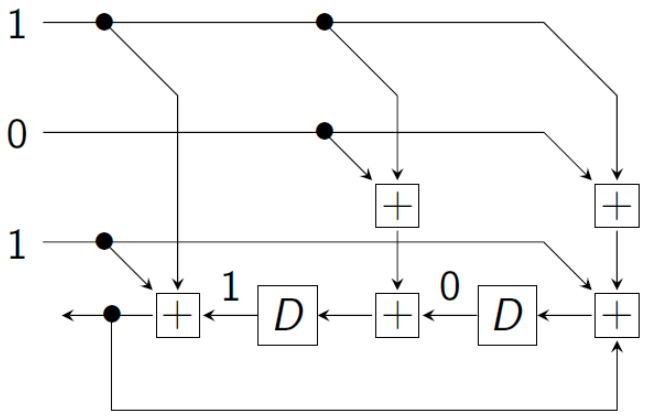
(4) $(s_0, \ldots, s_d) := (s_0, \ldots, s_d) + y_i^{(j)}(f_0^{(j)}, \ldots, f_d^{(j)})$

(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 $\{z_i\}_{i=0}^{\ell-1}$

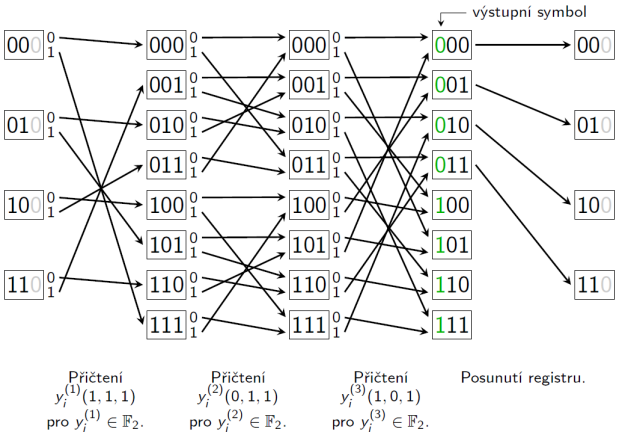


Na počátku máme $(s_0, s_1, s_2) = (\mathbf{1}, \mathbf{0}, 0)$. Podkroky pak jsou:

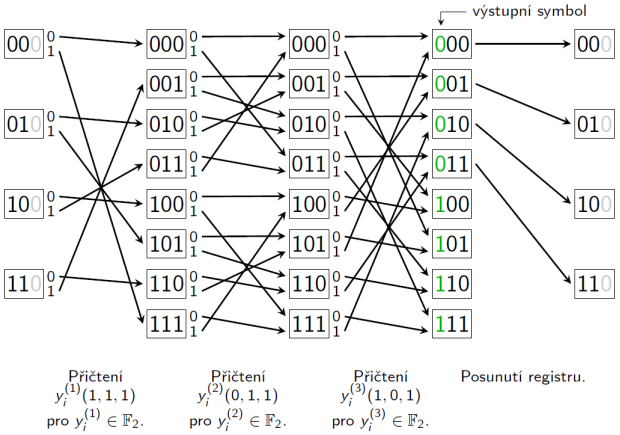
1. $(s_0, s_1, s_2) := (s_0, s_1, s_2) + \mathbf{1}(1, 1, 1) = (0, 1, 1)$ [řádek 4]
2. $(s_0, s_1, s_2) := (s_0, s_1, s_2) + \mathbf{0}(0, 1, 1) = (0, 1, 1)$ [řádek 4]
3. $(s_0, s_1, s_2) := (s_0, s_1, s_2) + \mathbf{1}(1, 0, 1) = (1, 1, 0)$ [řádek 4]
4. $(s_0, s_1, s_2) := (s_0, s_1, s_2) + 1(0, 0, 1) = (1, 1, 1)$ [řádek 6]

   $(s_0, s_1, s_2) := (s_1, s_2, 0) = (1, 1, 0)$ [řádek 7]

Rozvinutí modulů z Obrázku 18 tedy vypadá následovně. Pro modul s výstupem 0:



a pro modul s výstupem 1:



Trelážové moduly nahradíme jejich rozvinutými verzemi. Viterbiho algoritmus pak najde cestu rozvinutou treláží, která se nejvíce „podobá“ nosiči. Tento algoritmus umí zohlednit váhy jednotlivých změn v nosiči.

Ke každému prvku nosiči $x_i^{(j)}$ přiřazujeme distorzní funkci

$$\rho_i^{(j)} : \mathbb{F}_q \to \mathbb{R}\,.$$

Hodnota $\rho_i^{(j)}(a)$ udává, jak moc by přispělo nastavení $\_i^{(j)} = a$ ve stegoobjektu k celkové distorzi vyvolané vkládáním. Uvedeme si příklady distorzních funkcí

- Chceme-li Hammingovu metriku, definujeme

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

- Pro psaní na mokrý papír definujeme

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

Můžeme taky provést kombinaci těchto dvou metrik. Například pro nulové AC koeficienty dáme ∞ a pro ostatní použijeme Hammingovu.

- Pro vkládání při kvantizaci definujeme

$$\rho_i^{(j)}(a) = \text{(distorze při vložení } a\text{)} - \text{(distorze při zaokrouhlení)}.$$

Algoritmus (vkládání pomocí Viterbiho algoritmu)

VSTUP:

distorzní funkce prvků nosiče $\{\rho_i\}_{i=0}^{\ell-1}$

zpráva $\{z_i\}_{i=0}^{\ell-1}$ nad $\mathbb{F}_q$

parametry $f_j^{(k)} \in \mathbb{F}_q$ a $g_j \in \mathbb{F}_q$

VYSTUP:

stegoobjekt $\{\mathbf{y}_i\}_{i=0}^{\ell-1}$

ALGORITMUS:

- Inicializace

(1) for $\mathbf{s} \in \mathbb{F}_q^{d+1} \setminus \{\mathbf{0}\}$ do

(2) $\quad w[\mathbf{s}] := \infty$

(3) $w[\mathbf{0}] := 0$

- Dopředný průchod treláží

(4) for $i = 0, \dots, \ell - 1$ do

(5) $\quad$ for $j = 1, \dots, n$ do

(6) $\quad\quad$ for $\mathbf{s} \in \mathbb{F}_q^{d+1}$ do

(7) $\quad\quad\quad path_i^{(j)}[\mathbf{s}] := \arg \min_{a \in \mathbb{F}_q} w[\mathbf{s} - a(f_0^{(j)}, \dots, f_d^{(j)})] + \rho_i^{(j)}(a)$

(8) $\quad\quad\quad w'[\mathbf{s}] := \min_{a \in \mathbb{F}_q} w[\mathbf{s} - a(f_0^{(j)}, \dots, f_d^{(j)})] + \rho_i^{(j)}(a)$

(9) $\quad\quad w := w'$

(10) $\quad$ for $(s_0, \dots, s_{d-1}) \in \mathbb{F}_q^d$ do

(11) $\quad\quad w'[(s_0, \dots, s_{d-1}, 0)] := w[(z_i, s_0, \dots, s_{d-1}) - z_i(g_0, \dots, g_d)]$

(12) $\quad\quad$ for $s_d \in \mathbb{F}_q \setminus \{0\}$ do

(13) $\quad\quad\quad w'[(s_0, \dots, s_d)] := \infty$

(14) $\quad w := w'$

- Volba stavu, ve kterém končí nejlehčí cesta

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

- Rekonstrukce nejlhečí cesty zpětným průchodem treláže

(16) for $i = \ell - 1, \dots, 0$ do

(17) $\quad (s_0, s_1, \dots, s_d) := (z_i, s_0, \dots, s_{d-1}) - z_i(g_0, \dots, g_d)$

(18) $\quad$ for $j = n, \dots, 1$ do

(19) $\quad\quad y_i^{(j)} := path_i^{(j)}[(s_0, \dots, s_d)]$

(20) $\quad\quad (s_0, \dots, s_d) := (s_0, \dots, s_d) - y_i^{(j)}(f_0^{(j)}, \dots, f_d^{(j)})$

(21) return $\{y_i\}_{i=00}^{\ell-1}$

- $w[\mathbf{s}]$ je váha nejlehčí cesty, která vede do $\mathbf{s}$
- $path_i^{(j)}[\mathbf{s}]$ je poslední hrana nejlehčí cesty, která vede do $\mathbf{s}$ v j–tém kroku i-tého modulu