METODY KÓDOVÁNÍ, ŠIFROVÁNÍ A BEZPEČNOSTI DAT

URČENO PRO VZDĚLÁVÁNÍ V AKREDITOVANÝCH STUDIJNÍCH PROGRAMECH

RADIM FARANA

ČÍSLO OPERAČNÍHO PROGRAMU: CZ.1.07
NÁZEV OPERAČNÍHO PROGRAMU:
VZDĚLÁVÁNÍ PRO KONKURENCESCHOPNOST
OPATŘENÍ: 7.2
ČÍSLO OBLASTI PODPORY: 7.2.2

INOVACE VÝUKY INFORMATICKÝCH PŘEDMĚTŮ VE STUDIJNÍCH PROGRAMECH OSTRAVSKÉ UNIVERZITY

REGISTRAČNÍ ČÍSLO PROJEKTU: CZ.1.07/2.2.00/28.0245

OSTRAVA 2013

Tento projekt je spolufinancován Evropským sociálním fondem a státním rozpočtem České republiky

Recenzent: Doc. RNDr. František Koliba, CSc.

Název: Metody kódování, šifrování a bezpečnosti dat
Autor: Radim Farana
Vydání: první, 2013
Počet stran: 156

Jazyková korektura nebyla provedena, za jazykovou stránku odpovídá autor.

OBSAH

1 Vstup do problematiky

Cíl:

Tento učební text přibližuje studentům základní informace z oblasti kódování a šifrování dat. Učební text přibližuje matematické základy a algoritmy pro tvorbu kódů minimální délky, kontrolních a samoopravných kódů několika tříd. V druhé části jsou představeny základní šifrovací algoritmy jak pro steganografii, transpoziční i transkripční kryptografické systémy až po systémy s veřejným klíčem a související problematiky výtahu zprávy a elektronického podpisu.
Studenti získají přehled a orientaci v základních pojmech z oblasti kódování a šifrování dat.

Po jejím prostudování byste měli být schopni:

- Definovat cíl učebního textu.
- Znát obsah učebního textu.

Klíčová slova této kapitoly:

Data, kód, kódování, šifrování, algoritmus.

Doba potřebná ke studiu: 0,15 hodiny

Průvodce studiem

Tento učební text navazuje na předchozí studium základů kódování a komprese dat. Problematiku kódování rozvíjí do větší hloubky a přidává řadu náročnějších metod kódování. Protože jednotlivé stupně studia jsou nezávislé, nemůže text spoléhat na předchozí znalosti studentů a musí zopakovat i základní znalosti. Činí tak ale formou zúženou, aby vznikl prostor pro další rozvoj studia.
Učební text je sestaven podle standardů používaných pro elektronické učební texty včetně obvyklých grafických symbolů, ***polotučnou kurzívou*** *jsou psány hlavní pojmy, které je třeba znát, zatímco pouze* **tučné písmo** *slouží ke grafickému zvýraznění části textu. V uvozovkách jsou uváděny základní definice.*

Seminární práce
V průběhu studia modulu studenti vypracovávají seminární práci na zvolené téma z oblasti kódování a šifrování dat. Práci zpracovávají formou prezentace s komentářem pro přednášejícího (buď jako komentáře jednotlivých stránek nebo jako samostatný text). Ke zpracování musí použít nejméně tři nezávislé hodnověrné zdroje informace a na závěr prezentace zhodnotit kvalitu a nezávislost použitých zdrojů.

2 Teorie informace

Cíl:

Cílem celé kapitoly je představit základní pojmy z oblasti teorie informace, nutné pro pochopení následujícího textu.

Po jejím prostudování byste měli být schopni:

- Vymezit obsah vědních oborů kybernetika a informatika.
- Definovat základní pojmy z teorie informace.
- Definovat jednotku informace.
- Určit informační obsah zprávy.

Klíčová slova této kapitoly:

Informatika, kybernetika, informace, teorie informace, jednotka informace, zdroj zpráv, informační entropie, zpráva, sdělení, přenosový řetězec, kódové slovo, kód, minimální délka kódového slova, střední délka kódového slova, redundance.

Doba potřebná ke studiu: 5 hodin

Průvodce studiem

Tato kapitola vymezuje vědní obor informatika a definuje základní pojmy z oblasti teorie informace. Pro další pochopení textu je potřeba správně chápat definici jednotky informace a ostatní základní pojmy.
Vyhraďte si na studium této kapitoly pět hodin a věnujte pozornost všem příkladům, nepodlehněte dojmu, že vše je samozřejmé a dávno známé.

2.1 Informatika

informatika

Prvním problémem je samo o sobě vymezení obsahu předmětu ***informatika***. Mnozí autoři se ve vymezení jejího oboru značně různí. Často se setkáváme s názorem „Informatika je věda o sběru, přenosu a zpracování informace.“ Takto je však často definována také ***kybernetika***. Odtud pramení názor, že ***informatika*** je jen nový název pro ***kybernetiku***. Abychom obě vědní disciplíny oddělili, přikloňme se spíše k definici [**80**]:

kybernetika

„Informatika je věda o zpracování informace, zejména za pomoci automatizovaných prostředků.“

myšlení

Připomeňme si na tomto místě také pojem ***myšlení***, abychom zdůraznili jeho odlišnost od předmětu informatiky. Myšlení (jako specifická vlastnost druhu Homo sapiens) je specifickým, vysoce organizovaným příkladem zpracování informace, ale zpracování informace je neoddělitelným rysem každého živého organismu, včetně jednobuněčného.

informace
zpráva

Klíčovým pojmem, od kterého je odvozen název informatiky, je pojem ***informace***. K jejímu vysvětlení použijeme další, obdobně těžce vyjádřitelný pojem, ***zpráva***. ***Zprávu*** chápeme jako relaci mezi zdrojem a odběratelem, při které dochází k přenosu informace. Jejím hlavním atributem, nejpodstatnějším rysem, je právě skutečnost, že obsahuje nějakou informaci.
Informaci můžeme definovat různě, např.:

„Informací nazýváme abstraktní veličinu, která může být přechovávána v určitých objektech, předávána určitými objekty, zpracovávána v určitých objektech a použita k řízení určitých objektů. Jako objekt přitom chápeme živé organismy, technická zařízení nebo soustavy těchto prvků.“

Přitom platí:

1. Stejná zpráva může přinášet různou informaci různým odběratelům. Ať jsou to třeba hlášení stavu vody na českých tocích, synoptické mapy povětrnostní situace, a zejména tajné zprávy, kódy a šifry. Např. zpráva „**Nebe nad Španělskem je čisté.**“ byla běžně vysílána ve španělském rozhlase. Většina obyvatel ji přešla bez povšimnutí. Byla však dohodnutým heslem pro vystoupení frankistů proti republikánské vládě a zahájila těžké boje.
   Závěr: množství informace ve zprávě je závislé na příjemci.
2. Stejná zpráva může přinést stejnému příjemci různou informaci. Dejme tomu, že za války vedené Alfácií proti Betácii a Gamácii, mohli zajatci v Betácii odeslat cestou Červeného kříže jednu ze tří předtištěných pohlednic (což bylo všeobecně známo):
   I. Cítím se dobře, nic mi nechybí.
   II. Jsem zcela zdráv.
   III. Nebojte se o mne, nic nepotřebuji.

   Zajatci v Gamácii mohli odeslat jedině pohlednici:
   I. Cítím se dobře, nic mi nechybí.

   Jedna alfácká matka měla dva syny. Oba padli do zajetí. Jeden v Betácii, druhý v Gamácii. Oba poslali domů stejnou pohlednici. Zpráva z Betácie přesto obsahovala více informace, neboť tento syn měl na výběr více různých zpráv.
   Závěr: množství informace ve zprávě je závislé na zdroji.

teorie informace

Teorie informace je věda, která studuje množství informace ve zprávách, způsoby jejich kódování a přenášení. Vznikla v období II. světové války, zejména s rozvojem šifrování.

pravděpodobnost

Základním principem určování informačního obsahu zpráv je zjištění, že zpráva obsahuje tím více informace, čím menší je ***pravděpodobnost*** jejího výskytu.

Vycházíme přitom z axiomatické teorie pravděpodobnosti, která je definována např. ve [**21**]:

Každému jevu $A \subset E$ (množina všech přípustných jevů, jistý jev) je přiřazeno jako pravděpodobnost číslo P(A), přičemž platí následující axiomy:

a) pravděpodobnost je nezáporná, tj.
$$P(A) \geq 0; \quad (1)$$

b) pravděpodobnost sjednocení konečně mnoha nebo spočetně mnoha* vzájemně neslučitelných jevů $A_1 \subset E$, $A_2 \subset E$, ... je rovna součtu pravděpodobností těchto jevů, tj.
$$P(A1 \cup A2 \cup ...) = P(A1) + P(A2) + ...; \quad (2)$$

c) pravděpodobnost jistého jevu E je rovna 1, tj.
$$P(E) = 1. \quad (3)$$

2.2 Jednotka informace

jednotka informace

Máme-li měřit množství informace ve zprávě, potřebujeme k tomu jednotku. Jednotka informace byla ustanovena následovně:

„Jednotka informace je takové množství informace, které získáme potvrzením, že nastala jedna ze dvou stejně pravděpodobných možností."

bit

Jednotku informace označujeme často jako **[bit]**, fyzikálně je bezrozměrná (stejně jako pravděpodobnost). Brzy se stalo nepohodlným pracovat s tak malou jednotkou a začala se používat jednotka **[Byte]** (čti bajt), která obsahuje osm bitů. A další pojmy, jako slovo (word), poloslovo apod. vyjádřené v počtu Byte. Těmito pojmy se zatím zabývat nebudeme. Vznikly pro potřeby výpočetní techniky, vlastní teorie informace je nepotřebuje.

Byte

Specifickou otázkou jsou jména pro násobné jednotky. Vžilo se označení odvozené od jednotek používajících desítkovou číselnou soustavu. Např. kilobit by tedy měl obsahovat 1000 bitů. Ale používají se přitom násobky odvozené od mocnin dvou. Proto 1 kilobit = 1024 bitů. Aby byly tyto problémy odstraněny ustanovila IEC (International Electrotechnical Commission – mezinárodní standardizační organizace) v prosinci 1998 nová jména jednotek a předpon pro binární násobky používané ve zpracování dat a při přenosech dat:

---

* Množina se nazývá *spočetnou*, jestliže lze její prvky vzájemně jednoznačně přiřadit prvkům množiny přirozených čísel {1, 2, 3, ...}. Tato definice vychází z axiomu, že množina přirozených čísel je sice nekonečná, ale *spočetná*. Můžeme dokázat *spočetnost* množiny celých čísel i množiny racionálních čísel. Množina iracionálních čísel je již *nespočetná*, proto je *nespočetná* také množina reálných čísel.

Faktor	Přepona	Zkratka	Původ	Odvozeno
2^{10}	kibi	Ki	kilobinary $(2^{10})^1$	kilo $(10^3)^1$
2^{20}	mebi	Mi	megabinary $(2^{10})^2$	kilo $(10^3)^2$
2^{30}	gibi	Gi	gigabinary $(2^{10})^3$	kilo $(10^3)^3$
2^{40}	tebi	Ti	terabinary $(2^{10})^4$	kilo $(10^3)^4$
2^{50}	pebi	Pi	petabinary $(2^{10})^5$	kilo $(10^3)^5$
2^{60}	exbi	Ei	exabinary $(2^{10})^6$	kilo $(10^3)^6$

Uvedené předpony nejsou součást jednotek systému SI. Jak je patrné z tabulky, vznikly nové jednotky odvozením z předpon používaných v SI systému jednotek. Druhá část předpony byla nahrazena slabikou bi (ze slova binary). V současné době ale prakticky nejsou používány.

binary digit

shannon

nat

hartley

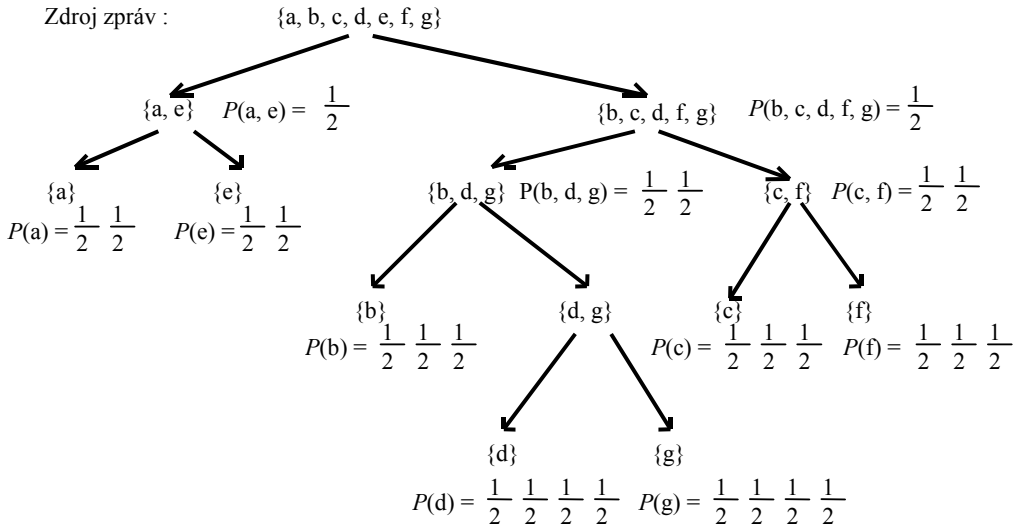
Označení **bit** má současně význam **binary digit**, a tak působí často nedorozumění, proto dnes používáme pro jednotku informace označení závislé na způsobu jejího určení podle vztahu (6), pro binární logaritmy je proto jednotkou **shannon**, [Sh]. Volba přirozených logaritmů ln se základem *e* dává jednotku **nat**, [nat]. Volba dekadických logaritmů lg dává jednotku **hartley**, [Hart] (viz norma IEC/ISO 80000, Díl 13).

zdroj informace

Příklad 2.1: Rozeberme ***zdroj informace*** vydávající celkem 7 zpráv {a, b, c, d, e, f, g}. Z hlediska pravděpodobnosti výskytu jednotlivých zpráv je můžeme rozdělit do dvou skupin o stejné pravděpodobnosti výskytu:

I. {a, e},
II. {b, c, d, f, g}.

Předpokládejme dále, že pravděpodobnosti {a} a {e} jsou stejné. Ve skupině II je stejně pravděpodobné, že zpráva patří do jedné z podskupin {b, d, g} nebo {c, f}, a dále podle obr. 1. Může se jednat např. o soubor hlášení s ustáleným textem.



Obr. 1. Pravděpodobnosti výskytů jednotlivých zpráv zdroje

Pravděpodobnost, že zpráva patří do skupiny I nebo II je stejná, tedy platí:

$$P(\text{a, e}) = P(\text{b, c, d, f, g}) = \frac{1}{2}, \tag{4a}$$

jestliže je zpráva ze skupiny I, pak je to {a} nebo {e} se stejnou pravděpodobností, takže:

$$P(\text{a}) + P(\text{e}) = \frac{1}{2}, \text{ a současně } P(\text{a}) = P(\text{e}), \text{ tedy } P(\text{a}) = P(\text{e}) = \frac{1}{4} = \left(\frac{1}{2}\right)^2. \tag{4b}$$

Podobně platí:

$$P(\text{b}) = P(\text{c}) = P(\text{f}) = \frac{1}{8} = \left(\frac{1}{2}\right)^3. \tag{4c}$$

$$P(\text{d}) = P(\text{g}) = \frac{1}{16} = \left(\frac{1}{2}\right)^4. \tag{4d}$$

Každé rozhodnutí, že zpráva patří do jedné ze dvou stejně pravděpodobných skupin, přináší informaci 1 bit. Zpráva {a} tedy obsahuje 2 bity informace, neboť bylo nutno učinit dvě rozhodnutí. Obecně pro náš příklad platí, že pravděpodobnost výskytu zprávy x je rovna:

$$P(x) = \left(\frac{1}{2}\right)^k, \tag{5}$$

kde je k – počet rozhodnutí při cestě rozhodovacím stromem k dané zprávě, ale také počet bitů informace, kterou zpráva nese, neboli její informační obsah.

2.3 Základní pojmy z teorie informace

informační obsah

Informační obsah zprávy x je roven:

$$k(x) = \log_2\left(\frac{1}{P(x)}\right) = -\log_2\left(P(x)\right) \quad [\text{Sh}]. \tag{6}$$

Odtud vyplývá, že pokud zkoumaný zdroj může dávat jen jednu zprávu, její pravděpodobnost je nutně rovna pravděpodobnosti jistého jevu, $P(\text{E}) = 1$, a informační obsah této zprávy je:

$$k = \log_2\left(\frac{1}{P(\text{E})}\right) = \log_2\left(\frac{1}{1}\right) = 0 \quad [\text{Sh}]. \tag{7}$$

Přenášet takovou zprávu nemá smysl.
Jestliže zdroj zpráv může dávat n různých zpráv x_i s pravděpodobností $P(x_i)$, $i = 1, 2, ..., n$, pak střední (vážené) množství informace ve zprávách z tohoto zdroje bude:

$$H = \sum_{i=1}^{n}\left(P(x_i)\cdot\log_2\left(\frac{1}{P(x_i)}\right)\right) \quad [\text{Sh/symbol}]. \tag{8}$$

informační entropie

Nazývá se ***informační entropie*** tohoto zdroje.

Informační entropie zdroje z obr. 1 je rovna:

$$H = \frac{1}{4}\cdot 2 + \frac{1}{4}\cdot 2 + \frac{1}{8}\cdot 3 + \frac{1}{8}\cdot 3 + \frac{1}{8}\cdot 3 + \frac{1}{16}\cdot 4 + \frac{1}{16}\cdot 4 \doteq 2{,}625 \quad [\text{Sh/symbol}]. \quad (9)$$

kódování

Zprávy z rozebíraného zdroje doposud označujeme {a, b, ..., g}. Ve skutečnosti mohou být reprezentovány nápisy libovolné délky. Protože známe jejich pravděpodobnosti výskytu, můžeme dohodnout způsob jejich ***kódování***. Přirozený způsob kódování vyplývá přímo z obr. 1. Budeme postupovat od vrcholu a popisovat cestu až ke zprávě, P – doprava, L – doleva, tedy:

a: LL,
e: LP,
b: PLL,
c: PPL,
f: PPP,
d: PLPL,
g: PLPP.

Tento zápis je velmi vhodný. Je možno jej interpretovat např. pomocí teček a čárek Morseovy abecedy apod. Obvykle však používáme znaky 0 a 1 binární abecedy, která je vhodnější pro interpretaci ve výpočetní technice. (Setkáme se také s označením L (Low – nízký signál) a H (High – vysoký signál), zejména u logických integrovaných obvodů). Provedeme-li pak transformaci L $\rightarrow$ 0, P $\rightarrow$ 1, dostaneme ***kód***:

kód

a: 00,
e: 01,
b: 100,
c: 110,
f: 111,
d: 1010,
g: 1011.

minimální délka kódového slova

Je zřejmé, že ***minimální délka kódového slova*** $N^*(x_i)$, vyjadřujícího kód zprávy o pravděpodobnosti $P(x_i)$ je rovna:

$$N^*(x_i) = -\log_2 P(x_i) = k(x_i) \ [\text{Sh}], \quad (10)$$

a je tedy rovna informačnímu obsahu zprávy.

V praxi však nebudou pravděpodobnosti výskytu zpráv vždy rovny celočíselným mocninám jedné poloviny. Zato kódová slova musí mít celistvý počet znaků. Ne vždy tedy budou délky kódových slov rovny minimálním délkám.

Pak pro kódová slova délky $N(x_i)$, $i = 1, 2, ..., n$ ze vztahu:

střední délka kódového slova

$$L = \sum_{i=1}^{n} P(x_i)\cdot N(x_i), \quad (11)$$

L nazýváme ***střední (váženou) délkou kódového slova***. A ze vztahu:

$$R = L - H, \quad (12)$$

redundance

R nazýváme ***redundancí (nadbytečností)*** daného způsobu kódování.

Poznámka k zamyšlení: Redundandnost kódu však má podstatný význam. V české abecedě je celkem 41 znaků, včetně přejatých znaků jako q, x apod.: a á b c č d ď e é ě f g h i í j k l m n ň o ó p q r ř s š t ť u ú ů v w x y ý z ž. Spisovná čeština obsahuje v nejrozsáhlejších slovnících (o devíti svazcích) asi 250 000 slov. Pokud by pravděpodobnost jejich výskytu byla konstantní, pak minimální délka slov bude:

$$N^*(x_i) = -\log_{41} P(x_i) = -\log_{41}\left(\frac{1}{250000}\right) \doteq 3{,}34697 \quad \text{[znak]}, \qquad (13)$$

takže všechna slova jsme schopni vyjádřit nejvýše čtyřmi znaky a ještě nám zbudou nevyužité kombinace znaků. Nebo opačně, chceme-li každé slovo vyjádřit právě pěti znaky, pak:

$$N^*(x_i) = 5 = -\log_z P(x_i) \Rightarrow z \doteq 12{,}0112, \qquad (14)$$

a k vyjádření všech slov nám postačí množina 12 různých znaků. Takové úvahy jsou zajisté scestné. Těžko si představíme použití slov jako „čšžýá“, „ggggg“ apod. Vysoká redundandnost jazyka má však pro naši schopnost dorozumění velký význam. Pomáhá nám překlenout rozdíly ve výslovnosti lidí (různá nářečí, polykání koncovek apod.) včetně různých vad výslovnosti (šišlání, ráčkování apod.). Např. Arabové využívají velké redundandnosti samohlásek v psaném textu tak, že je vůbec nezapisují. Aniž by přitom text ztratil na srozumitelnosti.
Obdobný význam má redundandnost kódu u technických systémů pro odstranění vlivu šumu v přenosovém kanále, přeslechů apod.

2.4 Přenos informace

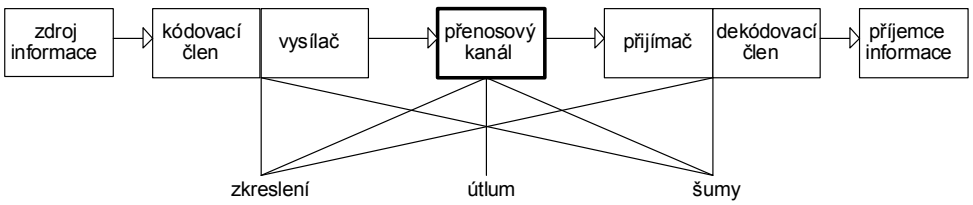
kód

Abychom mohli s informací pracovat, je třeba ji vyjádřit (reprezentovat) nějakým vhodným způsobem. K tomu využíváme různé ***kódy*** k vyjádření informace. Každý takový kód obsahuje množinu prvků (kódových slov), které reprezentují (zastupují) konkrétní elementární zprávy. Typickým příkladem je kódování jednotlivých znaků textu (např. dálnopisný kód CCITT, kód ASCII apod.). V tomto kódu se s informací přímo pracuje, proto ho obvykle označujeme jako ***abecedu*** (abecedu zdroje zpráv). Pojem kódování je totiž spojován spíše s přenosem informace. Této oblasti je věnována kapitola 2.

abeceda

Celou problematiku přenosu informace naznačuje obr. 2. Úplný ***přenosový řetězec*** tvoří zdroj informace – kódovací člen – vysílač – přenosový kanál – přijímač – dekódovací člen – příjemce informace. V dolní části jsou pak naznačeny hlavní nežádoucí vlivy, které na přenos informace působí.

přenosový řetězec



Obr. 2. Přenosový řetězec

přenosový kanál

spojitý přenosový kanál

diskrétní přenosový kanál

kvantování

bezšumový kanál

šumový kanál

odhalení chyby

odstranění chyby

bezpaměťový kanál

paměťový kanál

Podle způsobu práce, technické realizace a působení nepříznivých vlivů dělíme ***přenosové kanály*** na:

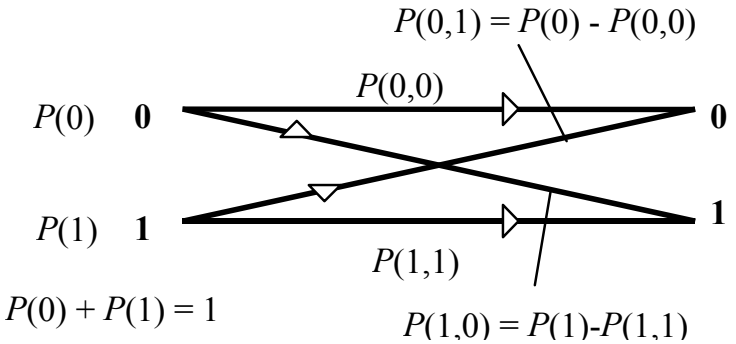
1. ***Spojité*** (v úrovni signálu) neboli ***analogové***. Přenosovým kanálem je přenášen signál, který může nabývat libovolných hodnot. Může přitom jít o ***signály spojité*** i ***diskrétní*** (z hlediska času). Těmito kanály se dále zabývat nebudeme.
2. ***Diskrétní*** (v úrovni signálu) neboli ***kvantované***. Přenosovým kanálem je přenášen signál, který nabývá hodnoty z množiny hodnot signálu. Takové signály vznikají ***kvantováním*** (diskretizací v úrovni) ze spojitých signálů. Z hlediska času může být signál spojitý nebo diskrétní.

Protože je možno všechny typy kanálů redukovat na diskrétní kanály s diskrétním časem, budeme se dále zabývat jen touto třídou přenosových kanálů. Tyto kanály se obvykle nazývají ***číslicové***. Z hlediska teorie informace rozdělujeme přenosové kanály podle pravděpodobnosti, že přijmeme skutečně vyslanou zprávu. Ideálním případem je ***bezšumový kanál***, přenášející informaci s naprostou jistotou, tedy pravděpodobností $P(x_i) = 1$. Běžné kanály tuto vlastnost nemají a vždy existuje nenulová pravděpodobnost, že se vyslaný znak změní ve kterýkoliv z možných znaků přijatých. Takové kanály označujeme jako ***šumové***.

Šumové kanály jsou v praxi častější a vyžadují speciální postupy zabezpečení přenosu informace sloužící k ***odhalení chyby*** či dokonce k ***odstranění*** zjištěné chyby. Z hlediska statistických závislostí chyb rozdělujeme šumové kanály na:

1. ***bezpaměťové***, kdy výskyt chyby v jednom znaku nijak neovlivňuje výskyt chyby v následujících znacích. Výskyt chyby je zcela náhodný a může být vyjádřen jednoduše pravděpodobností výskytu chyby, hovoříme tedy o ***chybách náhodných***.
2. ***paměťové***, kdy výskyt jedné chyby ovlivňuje chyby další. V takových kanálech mají chyby tendenci ke shlukování. Výskyt ***shlukových chyb*** je pro jejich rozpoznání a odstranění obvykle náročnější než výskyt několika náhodných chyb.

Nejčastěji budeme pracovat s bezpaměťovými kanály, u nichž bude pravděpodobnost změny jednoho kódového slova v jiné stejná pro všechna kódová slova. Takové kanály se nazývají symetrické. Rozložení pravděpodobnosti přenosu binárních signálů (z abecedy $Z_2 = \{0, 1\}$) je ukázáno na obr. 3.



Obr. 3. Symetrický binární kanál

U symetrického bezpaměťového šumového přenosového kanálu podle obr. 3 platí rovnost $P(1,0) = P(0,1)$. Nejčastěji budeme pracovat s kanály, u nichž předpokládáme také rovnost $P(0) = P(1) = 0{,}5$, tedy statisticky stejně pravděpodobný přenos obou znaků zdrojové abecedy. Pro popis vlastností kanálu nám pak postačí znalost pravděpodobnosti změny znaku $P(1,0)$ nebo $P(0,1)$, kterou označujeme také jednoduše jako ***pravděpodobnost výskytu chyby***.

pravděpodobnost výskytu chyby

Druhou významnou vlastností přenosového kanálu je počet znaků, které je možno jím přenést za jednotku času, tedy ***rychlost přenosu informace***. Její vyjádření přímo v počtu znaků zdrojové abecedy za sekundu je však nepraktické. Raději pracujeme s jejich informačním obsahem vyjádřeným v bitech. Např. při přenosu číslic desítkové soustavy a při předpokladu konstantní pravděpodobnosti jejich výskytu je informační obsah jedné číslice roven:

rychlost přenosu informace

$$k(x_i) = -\log_2 P(x_i) = -\log_2\left(\frac{1}{10}\right) \doteq 3{,}32 \text{bitů}\,. \qquad (15)$$

Kanál, který přenese 100 desítkových číslic za sekundu, přenese tedy množství informace rovné 332 bitů za jednu sekundu a rychlost přenosu je rovna 332 bit.s^{-1}. Neboli ***rychlost přenosu informace*** (v_p) je rovna množství informace přenesenému informačním kanálem za jednotku času.

modulační rychlost

V literatuře je možno se setkat s jinou definicí rychlosti přenosu informace, nazývanou ***modulační rychlost*** (v_m) nebo také ***telegrafní rychlost***, která je definována normou CCITT s jednotkou **[*Baud*]** (čti bód) označenou Bd, pojmenovanou podle tvůrce telegrafní abecedy, francouzského technika J. Baudota:

telegrafní rychlost

Baud

$$v_m = \frac{1}{i_{\min}} \quad [Bd]\,, \qquad (16a)$$

kde je $i_{\min}$ – minimální odstup dvou proudových změn při přenosu informace.

Z praktického pohledu je u binárních (dvoustavových) signálů $i_{\min}$ rovno délce přenosu jednoho bitu a jednotky ***Baud*** a ***bit.s*$^{-1}$**, mají shodný význam. To je zřejmě důvodem, proč se v poslední době používá téměř výhradně jednotka bit.s^{-1}. Rozdíly nastávají až při použití vícestavových číslicových signálů. U nich je celá skupina bitů vyjádřena jedním stavem signálu a hodnoty modulační rychlosti a rychlosti přenosu jsou různé, vztah mezi nimi je

bit.s^{-1}

$$v_p = v_m \cdot \log_2 m \,[\text{bit.s}^{-1}], \qquad (16b)$$

kde je v_p – rychlost přenosu,
v_m – modulační rychlost,
m – počet stavů vícestavového signálu.

Kontrolní otázky:

1. Jak je definován vědecký obor informatika?
2. Jak je definován pojem informace?
3. Jak je definována jednotka informace?
4. Jak určíme informační obsah zprávy?

Úkoly k textu:

1. Zdroj informace generuje 8 různých zpráv s četnostmi (10, 12, 25, 5, 10, 13, 15, 10). Jaká je informační entropie tohoto zdroje?
2. Přenosový kanál přenese 650 zpráv z tohoto zdroje za sekundu. Jaká je jeho přenosová rychlost?

Korespondenční úkol:

1. Vytvořte příklad zdroje zpráv s alespoň deseti různými zprávami, určete četnosti výskytu zpráv a spočítejte informační entropii tohoto zdroje.

Shrnutí obsahu kapitoly

V této kapitole jsme se seznámili se základními pojmy z oblasti teorie informace. Především je to sama definice pojmu informace, dále zpráva a její informační obsah. A samozřejmě definice potřebné jednotky informace. Dále jsme se seznámili s prvky přenosového řetězce, s vlastnostmi přenosových kanálů a rychlostí přenosu informace. Víme, že zprávu musíme pro přenos upravit do podoby sdělení a použít vhodný kód. Podrobně se kódováním bude zabývat následující kapitola.

3 Kódování

Cíl:

Cílem celé kapitoly je představit základní pojmy z oblasti teorie kódování, nutné pro pochopení základních principů tvorby kódů. Následuje vysvětlení konstrukce kódů minimální délky, používaných také pro kompresi dat. Dále jsou představeny principy nutné pro odhalení nebo opravu chyby při přenosu dat.

Po jejím prostudování byste měli být schopni:

- Definovat základní pojmy z oblasti kódování.
- Klasifikovat typy kódů a rozhodnout o jejich vhodném využití.
- Definovat a vysvětlit principy odhalování a opravy chyb při přenosu dat.

Klíčová slova této kapitoly:

Teorie kódování, prosté kódování, systematický kód, prefixový kód, blokový kód, Kraftova nerovnost, Mc Millanova věta, kód minimální délky, perfektní kód, Huffmanův kód, Shannon-Fanův kód, násobnost chyby, kontrolní kód, samoopravný kód, Hammingova vzdálenost, informační poměr.

Doba potřebná ke studiu: 2 hodiny

Průvodce studiem

Studium této kapitoly je nutné k pochopení základních pojmů z teorie kódování. Následně jsou představeny základní algoritmy konstrukce nejkratších kódů. Ty se používají všude tam, kde se snažíme minimalizovat délku kódových slov, zrychlit přenos informace apod. Kapitola končí základními principy odhalení a opravy chyb u blokových kódů.
Na studium této části si vyhraďte dvě hodiny. Zkuste si samostatně vyřešit příklady použití popsaných algoritmů tvorby minimálních kódů, abyste je byli schopni aplikovat a využít.

3.1 Základní pojmy z teorie kódů

teorie kódování

Teorie kódování se zabývá konstrukcemi kódů, zaměřenými na různé cíle (odstraňování chyb, zrychlení přenosu apod.) a studiem vlastností kódů.

zdrojový znak

zdrojová abeceda

kódový znak

kódová abeceda

Při přenosu nahrazujeme často znaky, kterými jsou zprávy zapsány (***zdrojové znaky*** ze ***zdrojové abecedy***), binárními symboly 0 a 1. Pokud je slov zdrojové abecedy více než dva, musíme používat slova vytvořená z binárních symbolů (***kódové znaky*** z ***kódové abecedy***), tedy posloupnost $b_1 b_2 ... b_k$, kde $b_i \in B$ pro $i = 1, 2, ..., k$ a $B = \{0, 1\}$. Hovoříme pak o ***kódovém slově*** délky k.

Množina všech kódových slov K(a), kde a jsou zdrojové znaky se nazývá jednoduše ***kód***. Význam mají jen ***prostá kódování***, kdy různým zdrojovým znakům odpovídají různá kódová slova.

prosté kódování

Každé kódování zdrojových znaků K: A→B můžeme rozšířit na kódování zdrojových zpráv, tedy posloupnosti slov v abecedě A,
$K^*(a_1 a_2 ... a_n) = K(a_1) \mid K(a_2) \mid ... \mid K(a_n)$.
Zprávu $a_1 a_2 ... a_n$ kódujeme znak po znaku. Tím vznikne zobrazení $K^*: A^* \rightarrow B^*$.

jednoznačné dekódování

prosté zobrazení

Podstatné je, abychom byli současně schopni ze zakódované zprávy získat zprávu zdrojovou. Kódování $K^*: A^* \rightarrow B^*$ je ***jednoznačně dekódovatelné***, jestliže ze znalosti zakódované zprávy $K^*(a_1 a_2 ... a_n)$ můžeme vždy jednoznačně určit zdrojovou zprávu $a_1 a_2 ... a_n$, tedy jestliže je kódování zpráv $K^*: A^* \rightarrow B^*$ ***prostým zobrazením*** (pro $x \neq y$ platí $f(x) \neq f(y)$).

Příklad 3.1: Máme za úkol kódovat informaci o stavu oblačnosti

jasno
polojasno
zataženo
déšť

Přitom víme, že nejčastěji je „jasno“, méně často „polojasno“ a nejméně často zbývající. Zvolme na ukázku dva příklady:

I		II	
jasno	0	jasno	0
polojasno	01	polojasno	01
zataženo	011	zataženo	011
déšť	111	déšť	101

V obou případech se jedná o prostá kódování, zprávu „jasno, jasno, polojasno, déšť“ zakódujeme:

I	II
0001111	0001101

Pokusme se nyní dekódovat zprávu 01101101:

I	II
Byť to na první pohled není zřejmé, zprávu lze dekódovat. Postupujeme „odzadu“. Jakmile narazíme na znak 0 nebo vyčerpáme maximální délku slova jedná se o začátek dalšího slova. Zpráva tedy zní: „zataženo, zataženo, polojasno“. Toto kódování zpráv je prostým zobrazením a můžeme ho použít. Při vyhodnocování zpráv ale musíme vždy přijmout celou zprávu až do konce.	Zkusme dekódovat zprávu rovněž „odzadu“. Dekódování však nebude jednoznačné. Můžeme získat zprávu: „zataženo, zataženo, polojasno“, ale také „polojasno, déšť, déšť“. Toto kódování zpráv tedy není prostým zobrazením a je nepoužitelné!

Výhodnější by bylo použít kódování, které bude dekódovatelné „od začátku“. Můžeme použít dva přístupy:

blokové kódování

a) *Blokové kódování* (délky n) je takové prosté kódování, při kterém mají všechna slova stejnou délku (a to n). Každé blokové kódování je jednoznačně dekódovatelné. Pokud známe zprávu K*(a_1 a_2 ... a_n), pak prvních n znaků tvoří kód zdrojového znaku a_1 atd.

prefixové kódování

b) *Prefixové kódování* je takové kódování, které je prosté a žádné kódové slovo není prefixem (začátkem) jiného kódového slova. Prefixové kódování je jednoznačně dekódovatelné. Ve zprávě najdeme nejmenší počet znaků, které tvoří kódové slovo, tyto znaky umažeme a pokračujeme.

Je zřejmé, že každé blokové kódování je současně prefixové. Použití prefixového kódování, které není blokové, nám zato umožňuje vytvořit kódy s malou redundancí, při jejich použití dosáhneme nejvyšší možné rychlosti přenosu zpráv. Umožňuje totiž zohlednit pravděpodobnost výskytu zprávy, tedy často se vyskytujícím zprávám přidělit krátká kódová slova.

Pro modelový příklad můžeme sestavit např. následující prefixové kódování:

jasno	0
polojasno	10
zataženo	110
déšť	111

Otázkou je, zda jsme schopni pro zadané požadavky sestavit prefixový kód. K rozhodnutí nám pomohou následující dvě věty:

Kraftova nerovnost

Prefixový kód sestrojený nad n-prvkovou kódovou abecedou s délkami kódových slov $d_1, d_2, \ldots, d_r$ existuje právě tehdy, když platí ***Kraftova nerovnost***, tj.

$$n^{-d_1} + n^{-d_2} + \cdots\cdots + n^{-d_r} \leq 1. \qquad (17)$$

.

Mc Millanova věta

Mc Millanova věta:
Každé jednoznačně dekódovatelné kódování splňuje Kraftovu nerovnost.

Z těchto dvou vět vyplývá, že každé jednoznačně dekódovatelné kódování je prefixové, ale pokud není, existuje jiné kódování nad stejnou kódovou abecedou s danými délkami kódových slov, které již prefixové bude.

Příklad 3.2: Máme za úkol sestavit binární prefixový kód číslic 0, 1, ..., 9 s délkami kódových slov $d_0 = d_1 = 2, d_2 = \cdots = d_7 = 3, d_8 = d_9 = 4$.

Kraftova nerovnost je rovna $2 \cdot \frac{1}{4} + 6 \cdot \frac{1}{8} + 2 \cdot \frac{1}{16} = \frac{11}{8} > 1$, tudíž prefixový kód s těmito délkami slov neexistuje.

Zvolíme-li $d_0 = d_1 = 2, d_2 = \cdots = d_7 = d_8 = d_9 = 4$, potom Kraftova nerovnost je $2 \cdot \frac{1}{4} + 8 \cdot \frac{1}{16} = 1$ a tedy takový prefixový kód existuje. Může vypadat například takto

Číslice	Kódové slovo
0	00
1	01
2	1000
3	1001
4	1010
5	1011
6	1100
7	1101
8	1110
9	1111

optimální kód

nejkratší kód

Naší snahou je však sestavit prefixový kód co nejkratší délky (***optimální kód***). Takové kódy se nazývají ***nejkratší kódy***. Pro jejich konstrukci vycházíme ze znalosti pravděpodobnosti výskytu zdrojových znaků $p_1, p_2, ..., p_n$ při délkách kódových slov $N_1, N_2, ..., N_n$ je pak střední délka kódového slova:

$$L = \sum_{i=1}^{n} p_i N_i \,. \tag{18}$$

Na dlouhou zprávu o m zdrojových znacích, pak potřebujeme přibližně $m \cdot L$ znaků kódových (binárních).

Příklad 3.3: Máme za úkol vytvořit kód pro vysílání zpráv složených ze znaků:

zdrojové znaky	+	–	*	/
pravděpodobnost výskytu	0,4	0,2	0,2	0,2
kódová slova	0	10	110	111

Střední délka slova je
$L = 1 \cdot 0{,}4 + 2 \cdot 0{,}2 + 3 \cdot 0{,}2 + 3 \cdot 0{,}2$.
Stejnou střední délku bude mít i kódování:

zdrojové znaky	+	–	*	/
kódová slova	00	01	10	11

nejkratší kód

Lepší kódování se nám najít nepodaří. Našli jsme tedy ***nejkratší kód***.

3.1.1 Huffmanova konstrukce nejkratšího kódu

Huffmanův kód

Nejkratší kód zkonstruoval v r. 1952 O. Huffman (čti hafmen), podle něj je často označován jako ***Huffmanův kód***. Zdrojové znaky uspořádáme podle nerostoucí posloupnosti pravděpodobnosti jejich výskytu. Pokud jsou jen dva, přiřadíme jim po řadě znaky 0 a 1.
V případě tří znaků pracujeme redukovanou abecedou a_1 a $a_{2,3}$ s pravděpodobnostmi výskytu p_1 a $p_{2,3} = p_2 + p_3$. Nejkratší kód redukované abecedy je

a_1	$a_{2,3}$
0	1

Rozdělením slova $a_{2,3}$ zpět na a_2 a a_3 dostaneme výsledný kód

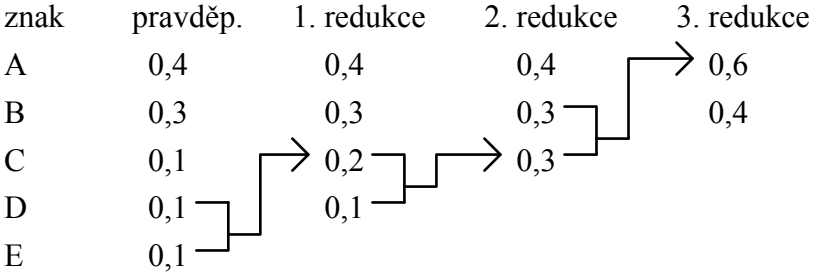
a_1	a_2	a_3
0	10	11

Obdobně postupujeme i při více zdrojových znacích.

Příklad 3.4: Sestrojme nejkratší kód pro zdrojovou abecedu:

A	B	C	D	E
0,4	0,3	0,1	0,1	0,1

Postup redukcí bude následující:



Zpětným postupem dostáváme tyto nejkratší kódy:

0,6 ... 0	0,4 ... 1	0,4 ... 1	0,4 ... 1
0,4 ... 1	0,3 ... 00	0,3 ... 00	0,3 ... 00
	0,3 ... 01	0,2 ... 010	0,1 ... 011
		0,1 ... 011	0,1 ...0100
			0,1 ... 0101

Výsledný kód je následující:

A	1
B	00
C	011
D	0100
E	0101

Jeho střední délka slova je:
$L = 0{,}4 + 2 \cdot 0{,}3 + (3 + 4 + 4) \cdot 0{,}1 = 2{,}1$ bitů .

Pozorný čtenář si jistě uvědomil, že Huffmanova konstrukce není jednoznačná. Závisí na tom, kam zařazujeme redukované znaky, pokud má více znaků stejnou pravděpodobnost výskytu.
Algoritmus na tvorbu Huffmanova kódu je možno účinně realizovat při využití binárních stromů. Podrobný popis algoritmu je k dispozici v literatuře, např. [2, 27].

3.1.2 Shannon-Fanova konstrukce nejkratšího kódu

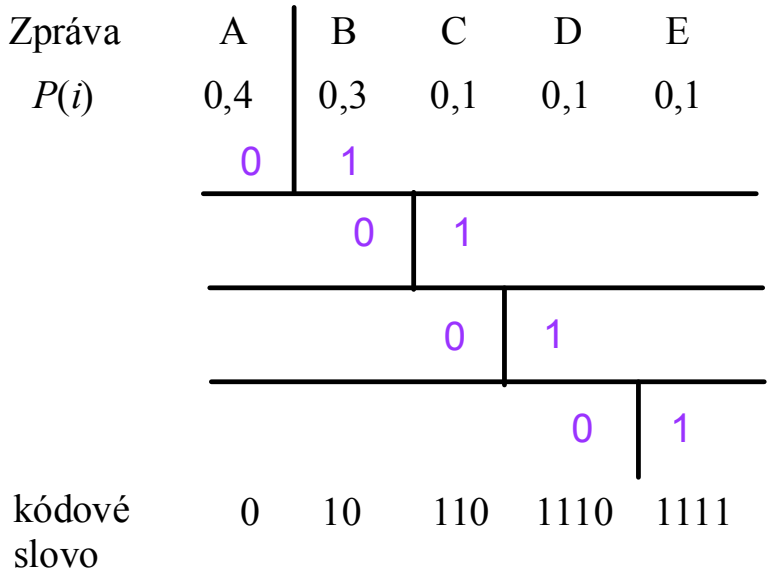
Shannon-Fanův kód

Odlišnou konstrukci vytvořili Claude Shannon a Robert Mario Fano. Kódová slova seřadíme podle pořadí jejich pravděpodobností. Následně je rozdělíme do dvou skupin, které se budou co nejméně lišit celkovou pravděpodobností. V každé skupině pokračuje dělení na skupiny až vzniknou jednoprvkové skupiny. Následně každé dvojici skupin přidělíme znaky 0 a 1 a sestavíme jednotlivá kódová slova.

Příklad 3.5: Sestrojme pomocí Shannon-Fanovy konstrukce nejkratší kód pro zdrojovou abecedu

A	B	C	D	E
0,4	0,3	0,1	0,1	0,1

Kódová slova již jsou seřazena podle pravděpodobnosti výskytu. Postup dělení skupiny bude následující



Pozorný čtenář si zajisté uvědomil, že Shannon-Fanova konstrukce není zcela jednoznačná. Závisí na tom, jak seřadíme znaky se stejnou pravděpodobností výskytu a v jakém pořadí budeme skupinám přidělovat znaky 0 a 1.

3.2 Bezpečnostní kódy

Jiným problémem je řešení přenosu informace, při kterém může docházet k ***chybám***. Tady naopak redundanci uměle zvyšujeme, abychom mohli chyby objevovat a nejlépe také odstraňovat.

bezpečnostní kód

3.2.1 Objevování chyb

objevování chyb

Pro objevování chyb je nutné, aby slova z množiny T^n byla rozdělena na ***kódová slova*** (K) a ***nekódová slova*** (T^n – K).

kódové slovo

Předpokládáme, že vysíláme kódová slova a přijímáme slova z množiny T^n. Pokud přijmeme nekódové slovo, pak jsme objevili chybu. Pokud je přijaté slovo kódové, pak buď nedošlo k chybě nebo jsme ji neobjevili.

nekódové slovo

Kód objevuje *t*-násobné chyby (chyby nejvýše v *t* znacích), jestliže při vyslání kódového slova a vzniku *t*-násobné chyby bude vždy přijato slovo nekódové.

K tomu definujeme ***Hammingovu vzdálenost***. Dvě slova mají Hammingovu vzdálenost rovnou počtu znaků, ve kterých se liší. Nejběžnější kódy pro objevování jednonásobných chyb jsou kódy kontroly parity. K binárnímu slovu přidáme znak tak, aby výsledné slovo mělo sudý počet jedniček (***sudou paritu***) nebo lichý (***lichou paritu***). Kód sudé parity je obvykle nazýván ***kódem celkové kontroly parity***. Jak bude ukázáno dále, kód liché parity má některé nepříjemné vlastnosti.

Hammingova vzdálenost

sudá parita

lichá parita

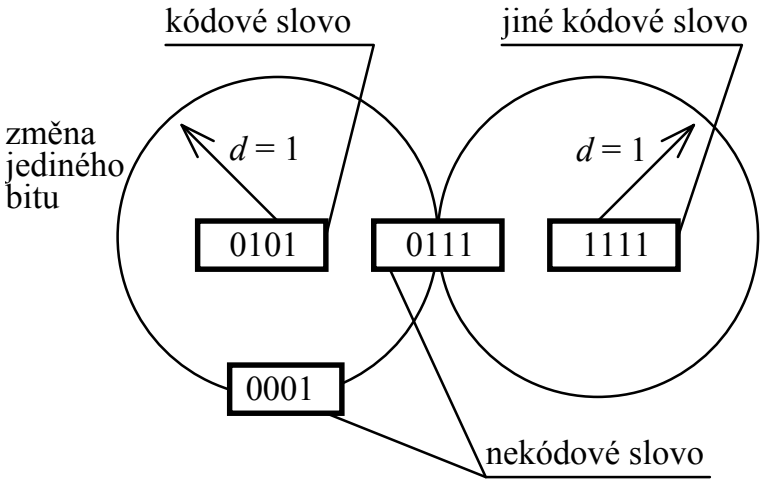
kód celkové kontroly parity

Výsledný kód skutečně objevuje všechny jednoduché chyby (vícenásobné objevovat nemusí).

(n, k)-kód

Např. ke tříznakovému kódu přidáme jeden bit paritní, výsledný kód bude mít čtyři znaky. Z toho tři znaky nesou informaci, jeden je nadbytečný, redundandní a slouží k zabezpečení. Hovoříme o ***kódu (4, 3)*** nebo obecně ***(n, k)-kódu***, kde je:
n – počet znaků,
k – počet informačních znaků.

Princip objevování chyb vysvětluje následující obr. 4.



Obr. 4. Princip objevování chyb

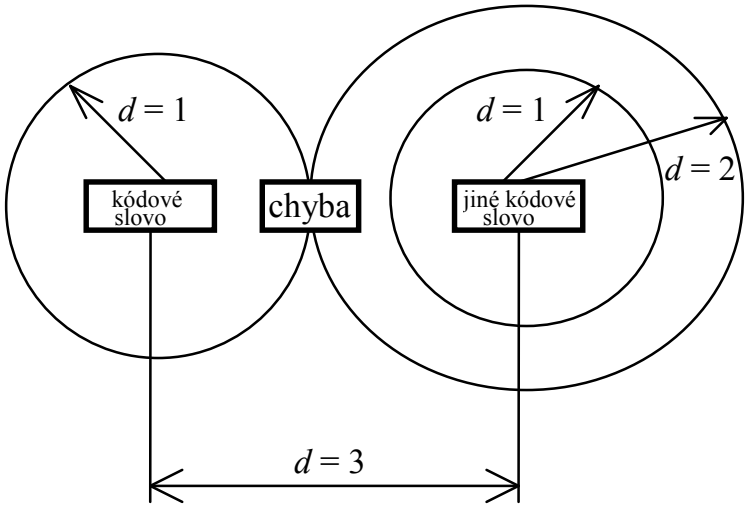
minimální vzdálenost

U kódu celkové parity mají všechna kódová slova Hammingovu vzdálenost alespoň 2, říkáme, že kód má ***minimální vzdálenost*** $d = 2$. Kód objevuje t-násobné chyby pro všechna $t < d$.

3.2.2 Opravování chyb

opravování chyb

Abychom mohli opravit t-násobnou chybu, musí platit $t < d/2$, jak vysvětluje následující obr. 5.



Obr. 5. Princip opravování chyb

Např. kód s minimální vzdáleností $d = 3$ je schopen opravit jednoduché chyby (a objevit dvojnásobné, ty by však byly opraveny nesprávně).

Příklad 3.6: jaké vlastnosti má kód „***dva z pěti***“, tedy binární kód obsahující všechna slova délky 5 znaků, z nichž jsou vždy dva znaky rovny 1?

kód “dva z pěti“

Všechna kódová slova jsou:

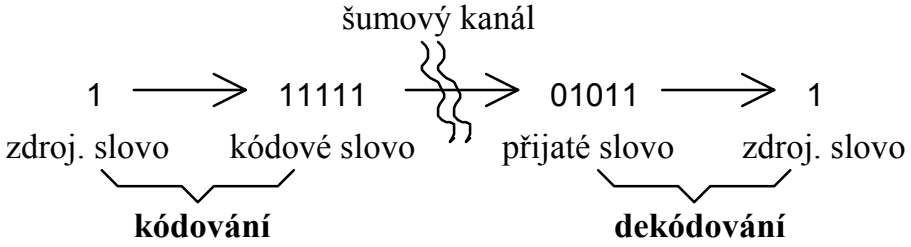
11000
10100
10010
10001
01001
00101
00011
00110
01100
01010

Minimální vzdálenost kódu je $d = 2$, protože Hammingova vzdálenost kódových slov je rovna 2 nebo 4. Potom tento kód objeví jednoduchou chybu a neopraví žádnou chybu.

Uvažujme například přijaté slovo 01011. Toto slovo není kódové a chybu objevíme, nevíme však, zda původním slovem bylo 01010, 00011 nebo 01001. Pokud budeme uvažovat dvojnásobné chyby, vyšleme například slovo 11000 a přijmeme 00110, dvojnásobnou chybu pak neobjevíme.

Pro velmi nekvalitní kanály je vhodný ***opakovací kód***. Každý znak vyšleme např. pětkrát, jedná se tedy o kód (5,1). Minimální vzdálenost kódu je $d = 5$. Kód objevuje 4-násobné chyby a opravuje 2-násobné chyby (Při výskytu 3-násobné či 4-násobné chyby provede kód samozřejmě opravu, ovšem na nesprávné kódové slovo). Dekódování nekódového slova se provádí „hlasováním“ – větší počet stejných znaků je uznán, např.:

opakovací kód

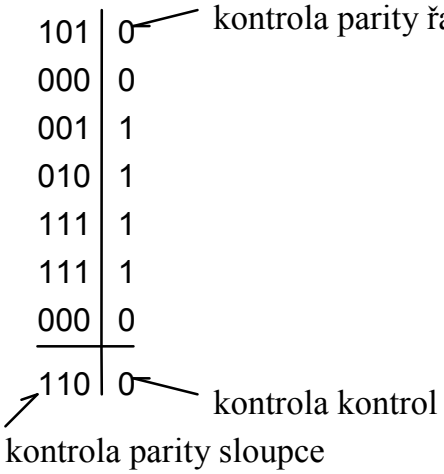


Velmi důležitý je ***kód dvourozměrné kontroly parity***, běžně používaný v počítačích. Informační znaky zapíšeme do matice typu (p, q). Každému řádku přidáme jeden symbol kontroly parity řádku, podobně každému sloupci kontrolu parity sloupce. Paritě sloupce parit řádků pak znak „kontrola kontrol“, volený tak, aby i parita výsledné matice byla sudá. Např. pro $p = 7$ a $q = 3$

kód dvourozměrné kontroly parity

máme 7.3 = 21 informačních znaků a 8.4 = 32 všech znaků, takže jde o kód délky 32. Nazývá se ASCII (32, 21)-kód.

Příklad kódového slova:



Tento kód opravuje jednoduché chyby. Taková chyba způsobí změnu parity v jednom řádku a jednom sloupci. Znak v jejich průsečíku můžeme opravit. Redundance kódu je ovšem poměrně vysoká. Při tvorbě kódů se snažíme najít kód, který má co nejkratší délku (nejmenší redundanci) a přitom opravuje co nejvíce chyb. V další kapitole proto přejdeme k lépe matematicky propracovaným kódům.

informační poměr

Pro hodnocení kódů se někdy používá ***informační poměr***, pro (*n*, *k*)-kód:

$$R = \frac{k}{n}. \tag{19}$$

Pro opakovací (5, 1)-kód vychází $R = \frac{1}{n} = 0{,}2$, pro ASCII (32, 21)-kód je $R = \frac{21}{32} = 0{,}65625$. ACSII (32, 21)-kód je tedy výhodnější než opakovací (5, 1)-kód. Než však tento závěr vyhlásíme, uvědomme si, prosím, že opakovací kód opraví dvojnásobnou chybu, zatímco ASCII kód pouze chybu jednoduchou.

Kontrolní otázky:

5. Co znamená pojem nejkratší kód?.
6. Popište Huffmanovu konstrukci nejkratšího kódu.
7. Popište Shannon-Fanovu konstrukci nejkratšího kódu.
8. Jaký je princip rozpoznání chyby při přenosu kódového slova?
9. Jaký je princip opravy chyby při přenosu kódového slova?

Úkoly k textu:

3. Pomocí Kraftovy nerovnosti rozhodněte, zda jsou kódy následující kódy jednoznačně dekódovatelné. Rozhodněte, zda jsou prefixové. Pokud jsou jednoznačně dekódovatelné, ale ne prefixové, zkonstruujte prefixové kódy se stejnými délkami kódových slov.

zpráva	K_1	K_2	K_3	K_4
A	0	0	0	2
B	10	02	1	01
C	11	1	210	02
D	101	12	211	202
E	100	20	212	201
F	111	21	222	222

4. Zdroj informace generuje 8 různých zpráv, ve sledované době byly zaznamenány jejich počty výskytů (10, 12, 25, 5, 10, 13, 15, 10). Sestavte kód minimální délky, určete jeho střední délku a redundanci daného způsobu kódování.
5. Jaké vlastnosti má kód celkové kontroly parity (sudá parita)?

Korespondenční úkol:

2. Vytvořte příklad zdroje zpráv s alespoň dvaceti různými zprávami, sestavte minimální kódy jak pomocí Huffmanovy, tak Shannon-Fanovy konstrukce, spočítejte informační entropii, střední délku kódového slova pro oba vytvořené kódy a porovnejte jejich redundanci.

Shrnutí obsahu kapitoly

V této kapitole jsme se seznámili se základními pojmy z teorie kódů. Známe základní podmínky dekódovatelnosti vytvořeného kódu. Umíme určit minimální délku kódového slova i střední délku kódového slova pro daný způsob kódování a určit jeho redundanci. Umíme použít Huffmanovu konstrukci nebo Shannon-fanovu konstrukci pro sestavení kódu minimální délky, což se nám bude hodit pro komprese dat. Chápeme základní principy důležité pro rozpoznání nebo dokonce opravu chyby vzniklé při přenosu kódového slova.

4 Lineární kódy

Cíl:

Cílem celé kapitoly je představit významnou třídu kódů – kódy lineární, někdy označované také za kódy maticové. Pochopení jejich principů je důležité pro zvládnutí následujících kapitol. Všechny další třídy kódů budou patřit ke kódům lineárním.

Po jejím prostudování byste měli být schopni:

- Definovat principy lineárních kódů.
- Rozpoznat zda je kód lineární.
- Sestavit lineární kód.

Klíčová slova této kapitoly:

Lineární kód, maticový kód, báze kódu, generující matice, kontrolní matice, systematický kód, syndrom slova, standardní dekódování, duální kód, samoduální kód, Hammingův kód, perfektní kód.

Doba potřebná ke studiu: 3 hodiny

Průvodce studiem

Studium této kapitoly je nutné k pochopení principů konstrukce kódů. Jsou představeny principy tvorby lineárních kódů, představeny systematické kódy, které se snadno dekódují. Kapitola končí třídou Hammingových kódů. je to významná třída kódů pro opravu jednoduchých chyb, které jsou perfektní, což znamená, že při požadovaných vlastnostech mají nejmenší možnou redundanci. Na studium této části si vyhraďte tři hodiny. Zkuste si samostatně vyřešit příklady tvorby lineárních kódů a jejich použití.

lineární kódy

Nejdůležitější třídou kódů pro nás budou ***lineární kódy***. To jsou kódy, u nichž je součet dvou kódových slov vždy opět kódovým slovem. Pro možnost manipulace s kódovými slovy je přitom vyjadřujeme jako řádkové vektory. Při definici těchto kódů se často používají matice, proto se také někdy označují jako ***maticové kódy***.

maticové kódy

Lineární kód tvoří lineární podprostor prostoru Z_2^n (n-rozměrný prostor binárních proměnných). Tento podprostor můžeme popsat jeho ***bází***. Např. kód celkové kontroly parity je (n, n – 1)-kód. Může mít bázi:

báze kódu

b_1 = 1000 ... 001

b_2 = 0100 ... 001

…

b_{n-1} = 0000 ... 011

generující matice kódu

Každý prvek báze je přitom sám kódovým slovem. Do báze se snažíme zařadit kódová slova s co nejmenším počtem jedniček, vzájemně lineárně nezávislá.
Tato báze b_1, b_2, ... b_k, (n, k)-kódu obsahuje kódová slova – vektory, které zapsány pod sebe tvoří ***generující matici kódu*** s n sloupci a k řádky.

$$\mathbf{G} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ . \\ . \\ . \\ \mathbf{b}_k \end{bmatrix}. \qquad (20)$$

Neboli matice **G** typu (k, n) je generující maticí lineárního kódu, jestliže platí:

- každý její řádek je kódovým slovem,
- každé kódové slovo je lineární kombinací řádků,
- řádky jsou lineárně nezávislé, to znamená, že hodnost matice **G** je k.

Např. kód celkové kontroly parity (sudá parita) (4, 3)-kód má generující matici:

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}. \qquad (21)$$

systematický kód

Pro další práci budou důležité zejména ***systematické kódy***. To jsou kódy, u nichž můžeme najít generující matici, která má v levé části jednotkovou matici.

$$\mathbf{G} = [\mathbf{E}|\mathbf{B}]. \qquad (22)$$

Kód celkové kontroly parity je tedy systematický.

Generující matici můžeme využít při kódování zpráv, neboť platí:

$$\mathbf{v} = \mathbf{z} \cdot \mathbf{G}, \qquad (23)$$

kde je **z** – slovo zdrojové abecedy,
v – slovo kódové abecedy,
G – generující matice lineárního kódu.

Například slovo zdrojové abecedy [100] zakódujeme:

$$\mathbf{v} = \begin{bmatrix}1 & 0 & 0\end{bmatrix} \cdot \begin{bmatrix}1 & 0 & 0 & 1\\ 0 & 1 & 0 & 1\\ 0 & 0 & 1 & 1\end{bmatrix} = \begin{bmatrix}1 & 0 & 0 & 1\end{bmatrix}, \text{ což je správné.}$$

kontrolní matice kódu

Větším problémem je dekódovatelnost zpráv, především kontrola, zda při přenosu nedošlo k chybě. U systematických kódů můžeme s výhodou použít ***kontrolní matici***.
Kontrolní matice lineárního kódu K je taková matice **H** z prvků abecedy T, pro kterou platí, slovo $\mathbf{v} = v_1 v_2 \ldots v_n$ je kódové, když splňuje podmínku:

$$\mathbf{H} \cdot \begin{bmatrix}v_1\\ v_2\\ .\\ .\\ .\\ v_n\end{bmatrix} = \begin{bmatrix}0\\ 0\\ .\\ .\\ .\\ 0\end{bmatrix}, \quad \mathbf{H} \cdot \mathbf{v}^{\mathrm{T}} = \mathbf{0}^{\mathrm{T}}. \tag{24}$$

Přitom platí věta:
Lineární kód s generující maticí $\mathbf{G} = [\mathbf{E} \mid \mathbf{B}]$ má kontrolní matici:

$$\mathbf{H} = \left[-\mathbf{B}^{\mathrm{T}} | \mathbf{E}'\right]. \tag{25}$$

Např. pro kód celkové kontroly parity (sudá parita)[#] je $\mathbf{H} = \begin{bmatrix}1 & 1 & 1 & 1\end{bmatrix}$.
Kontrolní matici lze s úspěchem použít pro objevování chyb. Pro každé slovo $\mathbf{v} = v_1 v_2 \ldots v_n$ definujeme slovo $\mathbf{s} = s_1 s_2 \ldots s_m$ předpisem:

$$\mathbf{H} \cdot \begin{bmatrix}v_1\\ v_2\\ .\\ .\\ .\\ v_n\end{bmatrix} = \begin{bmatrix}s_1\\ s_2\\ .\\ .\\ .\\ s_m\end{bmatrix}, \quad \mathbf{H} \cdot \mathbf{v}^{\mathrm{T}} = \mathbf{s}^{\mathrm{T}}. \tag{26}$$

syndrom slova

Slovo **s** se nazývá ***syndrom slova***. Je-li syndrom nenulový, pak došlo k chybě při přenosu. Syndromy lze využít k odhalování chyb poměrně rychlým algoritmem. Existuje dokonalejší, ale pomalejší postup – tzv. ***standardní dekódování***, popsané v další kapitole.

standardní dekódování

ekvivalentní systematický lineární kód

Ne všechny kódy jsou systematické. Naštěstí platí, že každý lineární kód je ***ekvivalentní systematickému lineárnímu kódu***, který nalezneme tak, že provedeme vhodnou permutaci sloupců v jeho generující matici.

[#] **Pozor !**
Kód liché parity není ***lineární***, např. součet dvou kódových slov 1101+1110 = 0011, což není kódové slovo. Nemá smysl snažit se sestavit jeho generující matici, natož matici kontrolní.

koktavý kód

Příklad 4.1: Máme definován (6, 3)-kód, u kterého je každý informační znak vyslán dvakrát. Takový kód se také nazývá ***koktavý kód*** délky 6. Tento kód je lineární a je reprezentován generující maticí **G** , kterou snadno sestavíme tak, že informačními znaky (sloupec 1, 3, 5) necháme probíhat jednotkové vektory a poté doplníme kontrolní znaky (sloupec 2, 4, a 6).

$$\mathbf{G} = \begin{bmatrix} 1 & b_{12} & 0 & b_{14} & 0 & b_{16} \\ 0 & b_{22} & 1 & b_{24} & 0 & b_{26} \\ 0 & b_{32} & 0 & b_{34} & 1 & b_{36} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}. \quad (27)$$

Koktavý kód není systematický. Vidíme však, že kdybychom v kódových slovech změnili pořadí symbolů tak, aby systematický byl, vytvořili bychom generující matici podobně, měla by pouze přeházené sloupce.

$$\mathbf{G'} = \begin{bmatrix} 1 & 0 & 0 & \beta_{14} & \beta_{15} & \beta_{16} \\ 0 & 1 & 0 & \beta_{24} & \beta_{25} & \beta_{26} \\ 0 & 0 & 1 & \beta_{34} & \beta_{35} & \beta_{36} \end{bmatrix}. \quad (28)$$

Vidíme, že první tři sloupce tvoří v matici systematického kódu jednotkovou submatici řádu 3. Označíme tedy druhou submatici jako **B**:

$$\mathbf{B} = \begin{bmatrix} \beta_{14} & \beta_{15} & \beta_{16} \\ \beta_{24} & \beta_{25} & \beta_{26} \\ \beta_{34} & \beta_{35} & \beta_{36} \end{bmatrix}. \quad (29)$$

Nyní vytvoříme kontrolní matici $\mathbf{H} = \left[-\mathbf{B}^{\mathrm{T}} \mid \mathbf{E'}\right]$, kde $-\mathbf{B}^{\mathrm{T}}$ je transponovaná záporná matice k matici **B** a **E'** je jednotková matice vhodného řádu (podle počtu řádků matice $-\mathbf{B}^{\mathrm{T}}$).

$$\left[-\mathbf{B}^{\mathrm{T}} \mid \mathbf{E'}\right] = \begin{bmatrix} -\beta_{14} & -\beta_{24} & -\beta_{34} & 1 & 0 & 0 \\ -\beta_{15} & -\beta_{25} & -\beta_{35} & 0 & 1 & 0 \\ -\beta_{16} & -\beta_{26} & -\beta_{36} & 0 & 0 & 1 \end{bmatrix}. \quad (30)$$

Pomocí kódování informačních znaků dostáváme z generující matice **G'** popis kódových slov kódu K':

$$\varphi(u_1, u_2, u_3) = (u_1 \quad u_2 \quad u_3 \quad u_1 \cdot \beta_{14} + u_2 \cdot \beta_{24} + u_3 \cdot \beta_{34} \quad u_1 \cdot \beta_{15} + u_2 \cdot \beta_{25} + u_3 \cdot \beta_{35} \quad u_1 \cdot \beta_{16} + u_2 \cdot \beta_{26} + u_3 \cdot \beta_{36}) \quad (31)$$

Nyní vynásobme nyní matici $\left[-\mathbf{B}^{\mathrm{T}} \mid \mathbf{E'}\right]$ transponovaným kódovým slovem u'^{T}

$$\left[-\mathbf{B}^{\mathrm{T}} \mid \mathbf{E}'\right]\cdot u'^{\mathrm{T}} = \begin{bmatrix} -u_1\cdot\beta_{14} - u_2\cdot\beta_{24} - u_3\cdot\beta_{34} + u_1\cdot\beta_{14} + u_2\cdot\beta_{24} + u_3\cdot\beta_{34} \\ -u_1\cdot\beta_{15} - u_2\cdot\beta_{25} - u_3\cdot\beta_{35} + u_1\cdot\beta_{15} + u_2\cdot\beta_{25} + u_3\cdot\beta_{35} \\ -u_1\cdot\beta_{16} - u_2\cdot\beta_{26} - u_3\cdot\beta_{36} + u_1\cdot\beta_{16} + u_2\cdot\beta_{26} + u_3\cdot\beta_{36} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

(32)

Z toho vyplývá, že matice $\left[-\mathbf{B}^{\mathrm{T}} \mid \mathbf{E}'\right]$ je kontrolní maticí kódu K', kterou jsme jednoduchým postupem zkonstruovali ze znalosti generující matice **G'**.

ekvivalentní systematický lineární kód

Vidíme, že platí: každý lineární kód je ***ekvivalentní systematickému lineárnímu kódu***, který získáme tak, že provedeme vhodnou permutaci sloupců v jeho generující matici.

Vrátíme-li se zpět ke koktavému kódu, můžeme provést permutaci sloupců (1, 5, 3, 4, 2, 6) a dostaneme matici **G'**, ve které můžeme dále vyměnit druhý a třetí řádek[#] a dostáváme generující matici ve tvaru $\mathbf{G} = \left[\mathbf{E} \mid \mathbf{B}\right]$ systematického lineárního (6, 3)-kódu K', který je ekvivalentní lineárnímu koktavému kódu délky 6.

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \sim \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}. \qquad (33)$$

Odtud snadno získáme kontrolní matici ekvivalentního systematického kódu K',

$$\mathbf{H}' = \begin{bmatrix} 0 & 0 & -1 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad (34)$$

a pokud provedeme zpětnou permutaci sloupců, získáme kontrolní matici původního koktavého kódu délky 6.

$$\mathbf{H} = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \qquad (35)$$

Na základě toho, že lineární kód je lineárním prostorem, můžeme definovat dualitu lineárních kódů.

duální kód

Nechť K je lineární (n, k)-kód, potom ***duální kód*** $\mathrm{K}^{\perp}$ takový, že platí $u' \in \mathrm{K}^{\perp} \Leftrightarrow \forall\, v' \in \mathrm{K} : u' * v' = 0$, kde je $*$ operace skalárního násobení vektorů.

[#] Řádky v generující matici tvoří bázová kódová slova a na jejich pořadí nezáleží

Platí, že duální kód $K^{\perp}$ lineárního (n, k)-kódu K je lineární(n, $n - k$)-kód a platí, že jejich generující a kontrolní matice jsou zaměněny, tedy $\mathbf{G} = \mathbf{H}'$ a $\mathbf{H} = \mathbf{G}'$.

Existuje také speciální skupina kódů, pro které platí, že $K = K^{\perp}$. Nazývají se ***samoduální kódy***. V jejich případě platí $\mathbf{G} = \mathbf{H}$.

samoduální kódy

4.1 Dekódování lineárních kódů

Nechť K je lineární (n, k)-kód. Potom kód K tvoří podgrupu aditivní grupy Z^n a odtud můžeme vytvořit faktorovou grupu Z^n / K, která tvoří ***grupu tříd kódu*** K tak, že pro každé $\mathbf{e} \in Z^n$ existuje třída $[\mathbf{w}]$ jejímž je reprezentantem

třídy kódu K

$$[\mathbf{ew}] = \mathbf{w} + \mathrm{K} = \{\mathbf{w} + \mathbf{v}; \forall\ \mathbf{v} \in \mathrm{K}\} \tag{36}$$

Z algebry víme, že jednotlivé třídy jsou buď disjunktní nebo jsou si rovny. Budeme tedy pracovat se systémem různých tříd, přičemž jako reprezentanty zvolíme vždy slovo dané třídy s nejmenší Hammingovou váhou. Reprezentanty budeme označovat znakem **e**

$$Z^n / \mathrm{K} = \left\{ [\mathbf{e}]; \mathbf{e} = \mathbf{w} \in Z^n \wedge \|\mathbf{e}\| = \|\mathbf{w}\| = \min_{v_i \in [w]} \|\mathbf{w}_i\| \right\}. \tag{37}$$

Dekódování pak probíhá podle předpisu

$$\partial(\mathbf{w}) = \mathbf{w} - \mathbf{e}_w, \text{ kde } \mathbf{e}_w \text{ je reprezentant třídy slova } \mathbf{w}. \tag{38}$$

Můžeme použít buď tabulku pro standardní dekódování nebo využít syndromy.

V prvním případě sestavíme ***standardní dekódovací tabulku,*** kde budou vypsány všechny třídy a jejich slova. V prvním sloupci bude vždy reprezentant třídy a řádky budou vzestupně uspořádány podle Hammingovy váhy jednotlivých reprezentantů. V prvním řádku bude tedy vždy třída s nulovým reprezentantem, která tvoří kód. na pozici i-tý řádek a j-tý sloupec bude takové slovo dané třídy, jehož rozdíl od kódového slova v j-tém sloupci je roven reprezentantu třídy.

standardní dekódování

standardní dekódovací tabulka

Příklad 4.2: Ukažme si dekódování na koktavém kódu délky 6. Všech slov délky 6 nad dvouprvkovou abecedou Z_2 je $2^6 = 64$, kód má 8 slov, potom standardní dekódovací tabulka bude mít 8 sloupců a 8 řádků – tříd[#].

	e_i							
K	000000	110000	001100	000011	110011	111100	001111	111111
	100000	010000	101100	100011	010011	011100	101111	011111
	001000	111000	000100	001011	111011	110100	000111	110111
	000010	110010	001110	000001	110001	111110	001101	111101
	101000	011000	100100	101011	011011	010100	100111	010111
	100010	010010	101110	100001	010001	011110	101101	011101
	000101	110101	001001	000110	110110	111001	001010	111010
	100110	010110	101010	100101	010101	011010	101001	011001

Dekódujeme potom snadno, najdeme dané slovo v tabulce a odečteme od něj reprezentanta. Měli bychom dostat slovo, které je kódovým slovem daného sloupce. Například

$$\partial(111000) = (111000) - (001000) = (110000)$$

Koktavý kód má minimální vzdálenost 2, potom objeví jednoduchou chybu, ale nemusí ji opravit správně.

Opravdu, stačí, když za reprezentanta třetího řádku místo 001000 zvolíme slovo této třídy stejné váhy 000100, potom třetí řádek by byl nahrazen tímto

000100	110100	001000	000111	110111	111000	001011	111011

a potom

$$\partial(111000) = (111000) - (000100) = (111100).$$

dekódování pomocí syndromů

Druhým způsobem je ***dekódování pomocí syndromů***. Celou standardní tabulku budeme redukovat na sloupec reprezentantů a jejich syndromů a využijeme vlastnost, že všechna slova dané třídy mají shodný syndrom.

e_i	$s_{e_i}^T$
000000	000
100000	010
001000	100
000010	001
101000	110
100010	011
000101	101
100110	111

[#] Při vytváření tabulky můžeme postupovat tak, že začnete nulovým reprezentantem a vypíšete kódová slova, poté pokračujete reprezentanty váhy 1, tedy 100...0, 010...0, až 000...1, potom váhy 2 atd., dokud nevygenerujeme celou tabulku. Přitom pokaždé, když se v řádku objeví slovo, které již bylo dříve uvedeno, tvoříme třídu shodnou s nějakou dříve uvedenou. Daný řádek smažeme a postupujeme dále.

Dekódujeme tak, že přijmeme slovo *w*, určíme jeho syndrom s_w, ten najdeme v tabulce a od slova *w* odečteme reprezentanta, který má shodný syndrom. Například přijmeme slovo (111000), potom jeho syndrom je

$$\mathbf{H}\cdot(111000)^{\mathrm{T}}=\begin{bmatrix}0&0&1&1&0&0\\1&1&0&0&0&0\\0&0&0&0&1&1\end{bmatrix}\cdot[111000]^{\mathrm{T}}=[100]^{\mathrm{T}}\neq\mathbf{o}^{\mathrm{T}} \qquad (39)$$

a odtud je

$$\partial(111000)=(111000)-(00100)=(110000).$$

4.2 Hammingovy kódy

Hammingovy kódy jsou speciální skupinou lineárních kódů. Opravují jednoduché chyby, neobyčejně snadno se dekódují a jsou ***perfektní*** (mají nejmenší možnou redundanci).

Hammingův kód

perfektní kód

Hammingův binární kód s *m* kontrolními znaky (*m* = 2, 3, 4, ...) má délku $2^m - 1$, takže vzniká (3, 1)-kód, (7, 4)-kód, (15, 11)-kód, atd. Binární kód se nazývá Hammingův, jestliže má kontrolní matici, jejíž sloupce jsou všechna nenulová slova dané délky a žádné z nich se neopakuje. Na uspořádání sloupců nezáleží. Můžeme použít např. binární rozvoj čísel 1, 2, ..., $2^m - 1$.

Pro *m* = 2 dostaneme matici

$$\mathbf{H}=\begin{bmatrix}0&1&1\\1&0&1\end{bmatrix}. \qquad (40)$$

Matice **H** je kontrolní maticí opakovacího kódu délky 3. Ten skutečně opravuje jednoduché chyby. Pro *m* = 3 dostaneme (7, 4)-kód s kontrolní maticí:

$$\mathbf{H}=\begin{bmatrix}0&0&0&1&1&1&1\\0&1&1&0&0&1&1\\1&0&1&0&1&0&1\end{bmatrix}. \qquad (41)$$

Abychom našli jeho generující matici, přemístíme sloupce do tvaru $\mathbf{H}=\left[-\mathbf{B}^{\mathrm{T}}\mid\mathbf{E}'\right]$. Např. vyměníme první a poslední, druhý a předposlední, čtvrtý a pátý sloupec. Dostaneme:

$$\mathbf{H}' = \left[\begin{array}{cccc|ccc} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{array}\right]. \tag{42}$$

Generující matice tohoto kódu bude $\mathbf{G} = [\mathbf{E} \mid \mathbf{B}]$

$$\mathbf{G}' = \left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{array}\right]. \tag{43}$$

Generující matici původního kódu získáme zpětným přehozením sloupců:

$$\mathbf{G} = \left[\begin{array}{cccc|ccc} 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \end{array}\right]. \tag{44}$$

Z hlediska kódování je výhodnější **G'**, neboť jde o systematický Hammingův kód. Z hlediska dekódování je výhodnější tvar **G**, či spíše **H**.
Dekódování Hammingova kódu využívá syndromu **s**. Jestliže sloupce matice **H** jsou uspořádány tak, že tvoří binární rozvoje čísel 1, 2, ..., $2^m - 1$, přijmeme vektor $\mathbf{v} = v_1 v_2 \ldots v_n$, určíme jeho syndrom **s**, kde:

$$\mathbf{s}^T = \mathbf{H}\mathbf{v}^T. \tag{45}$$

rozšířený Hammingův kód

Slovo **s** je binárním zápisem čísla *i* a my změníme *i*-tý znak přijatého slova. Hammingův kód má minimální vzdálenost $d = 3$, opravuje tedy jednoduché chyby. ***Rozšířený Hammingův kód*** (doplněný o paritní bit) pak má vzdálenost $d = 4$, takže odhaluje trojnásobné chyby.

Kontrolní otázky:

10. Jaké vlastnosti má lineární kód?.
11. Jak se sestavuje generující matice lineárního kódu?
12. Jaké vlastnosti má systematický kód?
13. Jaký je vztah mezi generující a kontrolní maticí systematického kódu?

Úkoly k textu:

6. Lineární kód obsahuje kódová slova (0000, 0011, 0101, 0110, 1001, 1010, 1100, 1111), sestavte jeho generující a kontrolní matici.
7. Pro binární kód celkové kontroly parity délky 4 sestavte standardní dekódovací tabulku a tabulku pro dekódovaní pomocí syndromů. Dekódujte slova 1011, 1110, 1111.

Shrnutí obsahu kapitoly

V této kapitole jsme se seznámili s nejdůležitější třídou kódů – lineárními kódy. Umíme sestavit generující matici lineárního kódu, určit kontrolní matici a použít ji pro odhalení chyby při přenosu. Umíme sestavit generující a kontrolní matici Hammingova kódu pro opravu jednoduchých chyb.

5 Reedovy-Mullerovy kódy

Cíl:

Cílem celé kapitoly je představit významnou třídu kódů Reedovy-Mullerovy kódy. Patří mezi lineární kódy, ale jejich konstrukce využívá booleovské funkce a booleovské polynomy. Tím se poněkud vymykají z této oblasti. jsou významné velmi jednoduchým dekódováním. Použila je např. sonda Mariner 9 pro vysílání fotografií z Marsu. Jsou to nejstarší kódy opravující volitelný počet chyb.

Po jejím prostudování byste měli být schopni:

- Pracovat s booleovskými funkcemi a booleovskými polynomy.
- Definovat základní typy Reedových-Mullerových kódů a jejich možnosti.
- Sestavit Reedův-Mullerův kód.

Klíčová slova této kapitoly:

Booleovská funkce, booleovský polynom, stupeň booleovského polynomu, Reedův-Mullerův kód, R-M kód, kódování R-M kódů, dekódování R-M kódů, opakovací kód, kód $R(0, m)$, kód $R(1, m)$, kód $R(r, m)$.

Doba potřebná ke studiu: 4 hodiny

Průvodce studiem

Studium této kapitoly je nutné k pochopení principů konstrukce RM-kódů. Použití booleovských funkcí a booleovských polynomů je využitelné i při optimalizaci logických obvodů a systémů logického řízení.
Na studium této části si vyhraďte čtyři hodiny. Zkuste si samostatně vyřešit příklady tvorby RM- kódů a jejich použití.

5.1 Booleovské funkce

Booleovská funkce

Booleovské funkce nabývají hodnot 0 a 1, stejně jako jejich proměnné. Booleovská funkce m proměnných je předpis, který m-tici $x_1, x_2, \ldots, x_m$ přiřazuje hodnotu $f(x_1, x_2, \ldots, x_m) = 0$ nebo 1. Předpis f tedy představuje zobrazení množiny Z_2^n (kde $n = 2^m$) do množiny Z_2. Toto zobrazení zapisujeme buď pravdivostní tabulkou nebo jako ***booleovské polynomy (mnohočleny)***.

Booleovský polynom

Příklad booleovské funkce tří proměnných $f(x_1, x_2, x_3)$:

x_1	0	1	0	1	0	1	0	1
x_2	0	0	1	1	0	0	1	1
x_3	0	0	0	0	1	1	1	1
f	0	1	1	0	1	0	0	1

Libovolnou booleovskou funkci $f(x_1, x_2, \ldots, x_n)$ můžeme chápat jako binární slovo délky 2^m $f_0 f_1 \ldots f_{2^m-1}$, kde pro každé číslo $j = 0, 1, \ldots, 2^m\text{-}1$ je $f_j = f(j_1, j_2, \ldots, j_n)$, jestliže má j binární rozvoj $j_n j_{n-1} \ldots j_1$ (Oproti zvyklostem bereme binární rozvoj čísla j pozpátku).
Například $f_0 = f(0, 0, \ldots, 0)$, $f_1 = f(1, 0, \ldots, 0)$, $f_2 = f(0, 1, \ldots, 0)$, $f_3 = f(1, 1, \ldots, 0)$, …
Mezi booleovské funkce patří také konstantní funkce ***o*** = 000…0 a ***1*** = 111…1 a také každá z n proměnných x_i je sama booleovskou funkcí $f(x_1, x_2, \ldots, x_n) = x_i$. Např. pro $m = 3$ je
$x_1 = 01010101$, $x_2 = 00110011$.

Na booleovských funkcích m proměnných jsou zavedeny obvyklé logické operace:

název	označení	= 1, právě když
f a g (logický součin)	fg	$f = 1$ a současně $g = 1$
f vel g (binární součet)	$f + g$	$f = 1$ nebo $g = 1$, ale ne obě
f nebo g (logický součet)	$f + g + fg$	$f = 1$ nebo $g = 1$ nebo $f = g = 1$
negace f	$\bar{f}$	$f = 0$

Funkce ovšem představují vícebitové vektory. Pak platí uvedené definice pro všechna f_i a g_i, představující stejnolehlé bity vektorů, např. Pro $f = 0101$ a $g = 0011$ platí:
$fg = 0001$
$f + g = 0110$
$f + g + fg = 0111$
$\bar{f} = 1010$

Mezi logickými operacemi existují jednoduché vztahy, např.:
$\bar{f} = 1 + f$ - negace tedy může být vyjádřena pomocí binárního součtu,
$ff = f$ - nikdy nebudeme potřebovat vyšší exponent než 1,
- operaci logického součtu (OR) můžeme realizovat pomocí logických součinů a binárního sčítání: f OR $g = f + g + fg$.

- booleovské funkce považujeme za prvky prostoru Z_2^n (kde $n = 2^m$) a logický součet se pak shoduje se sčítáním v tomto prostoru.
- výrazy 1 a x_i jsou booleovské funkce, všechny booleovské funkce můžeme realizovat jejich sčítáním a násobením.

Například pro funkci $f = 1011$ můžeme zvolit pro snadnější reprezentaci (menší počet jedniček) negaci funkce $\bar{f} = 0100$. Ta nabývá 1 právě když $x_1 = 1$ a $x_2 = 0$, takže:

$$\bar{f} = x_1\bar{x}_2 = x_1(1 + x_2) = x_1 + x_1x_2 . \qquad (46)$$

Výraz pro funkci f je pak ve tvaru:

$$f = 1 + \bar{f} = 1 + x_1 + x_1x_2$$

Tento výraz se nazývá ***booleovský polynom (mnohočlen)***. Každý booleovský polynom je tedy součtem členů:

$$x_1^{i_1} x_2^{i_2} \ldots x_m^{i_m} , \qquad (47)$$

kde mocnitelé i_1, i_2, … i_m jsou buď 1 nebo 0 (pokud se daná proměnná ve výrazu nevyskytuje). Například pro $m = 3$ máme:

$$x_1^1 x_2^0 x_3^0 = x_1 ,$$

$$x_1^1 x_2^0 x_3^1 = x_1x_3 , \qquad (48)$$

$$x_1^0 x_2^0 x_3^0 = \mathbf{1} ,$$

Booleovské polynomy můžeme zapisovat ve tvaru:

$$f(x_1, x_2, \ldots, x_m) = \sum_{i=0}^{2^m - 1} q_i x_1^{i_1} x_2^{i_2} \ldots x_m^{i_m} . \qquad (49)$$

Kde koeficient q_i je buď 0 nebo 1 a číslo i má binární rozvoj $i_m \ldots i_2 i_1$ (který čtením pozpátku určuje mocnitele). Například polynom $f = x_1 + x_1x_2$ je ve tvaru:

$$f = q_0 + q_1x_1 + q_2x_2 + q_3x_1x_2 \text{ kde } q_0 = q_2 = 0 \text{ a } q_1 = q_3 = 1. \qquad (50)$$

stupeň booleovského polynomu

Stupeň booleovského polynomu f je největší počet proměnných, které se vyskytují v některém sčítanci polynomu f. Nulová konstantní funkce **0** představuje booleovský polynom stupně -1, funkce **1** je booleovský polynom stupně 0 a booleovský polynom f, který má stupeň $k \geq 1$, má maximální počet sčítanců k. Například polynom $1 + x_1 + x_3$ má stupeň 1, polynom $x_1x_2x_3$ má stupeň 3.

Jestliže má booleovská funkce tří proměnných $f(x_1, x_2, x_3)$ binární zápis $f_0f_1 \ldots f_7$, potom první polovina tohoto slova (neboli slovo f_0f_1 f_2f_3) představuje binární zápis booleovské funkce dvou proměnných $f(x_1, x_2, 0)$ a druhá polovina (neboli slovo f_4f_5 f_6f_7) představuje binární zápis booleovské funkce dvou proměnných $f(x_1, x_2, 1)$. Například pro funkci:

$$\begin{aligned} f(x_1, x_2, x_3) &= x_1 + x_3 \\ &= 01010101 + 00001111 \\ &= 0101 \mid 1010 \end{aligned} \tag{51}$$

Pro každou booleovskou funkci $m + 1$ proměnných $f(x_1, x_2, ..., x_m, x_{m+1})$ platí:

$$f = f(x_1, x_2, ..., x_m, 0) + [f(x_1, x_2, ..., x_m, 0) + f(x_1, x_2, ..., x_m, 1)]x_{m+1}. \tag{52}$$

Na základě těchto dvou pravidel můžeme snadno převádět booleovské funkce na booleovské polynomy.

Například booleovskou funkci $f = 1011$ převedeme na booleovský polynom:

$$f = 10 + (10 + 11)x_2 = 10 + 01x_2. \tag{53}$$

Stejným postupem je $10 = 1 + (1 + 0)x_1 = 1 + x_1$ a $01 = 0 + (0 + 1)x_1 = x_1$, takže:

$$f = 1 + x_1 + x_1x_2. \tag{54}$$

Podobně pro booleovskou funkci tří proměnných $f = 11000111$ platí:

$f = 1100 + (1100+0111)x_3 = 1100 + 1011x_3 = 1 + x_2 + (1 + x_1 + x_1x_2)x_3 =$
$= 1 + x_2 + x_3 + x_2x_3 + x_1x_2x_3$

protože $1100 = 11 + (11 + 00)\, x_2 = 11 + 11x_2 = 1 + x_2$,
a to neboť $11 = 1 + (1 + 1)x_1 = 1 + 0x_1 = 1$.

a podobně $1011 = 10 + (10 + 11)x_2 = 10 + 01x_2 = 1 + x_1 + x_1x_2$,
a to neboť $10 = 1 + (1 + 0)x_1 = 1 + x_1$,
a dále $01 = 0 + (0 + 1)x_1 = x_1$.

Dále využijeme množiny $M(i)$. Pro každé číslo $i = 0, 1, \ldots, 2^m\text{-}1$ označujeme jako $M(i)$ množinu těch čísel $j \le i$, která má ve svém binárním rozvoji jedničky jen tam, kde je má číslo i. V binárním zápisu tedy bude:

$$M(i_m ... i_2 i_1) = \left\{ j_m ... j_2 j_1 \,\middle|\, \text{z } j_k = 1 \text{ plyne } i_k = 1 \text{ pro } k = 1,2,...,m \right\}. \tag{55}$$

Například
$M(0) = \{0\}$, $M(1) = \{0, 1\}$, $M(2) = \{0, 2\}$, $M(3) = \{0, 1, 2, 3\}$,
Vidíme, že platí $M(2^i) = \{0, 2^i\}$

Každá booleovská funkce $f = f_0 f_1 ... f_{2^m-1}$ je polynomem:

$$f = \sum_{i=0}^{2^m-1} q_i x_1^{i_1} x_2^{i_2} ... x_m^{i_m}, \tag{56}$$

(kde číslo i má binární rozvoj $i_m \ldots i_1$) s koeficienty

$$q_i = \sum_{j \in M(i)} f_j . \tag{57}$$

Příklad 5.1: Pro booleovskou funkci f=1011 máme
$q_0 = f_0 = 1$
$q_1 = f_0 + f_1 = 1$
$q_2 = f_0 + f_2 = 0$
$q_3 = f_0 + f_1 + f_2 + f_3 = 1$
takže platí
$f(x_1, x_2) = 1 + x_1 + x_1x_2$.

Booleovské polynomy o jediném sčítanci, tedy funkce:
1,
$x_1, x_2, \ldots, x_m$,
x_ix_j pro $i, j = 1, \ldots, m$,
…
$x_1x_2\ldots x_m$.
tvoří bázi lineárního prostoru Z_2^n (kde $n = 2^m$), protože každé slovo v Z_2^n je součtem některých z těchto funkcí. Lineární nezávislost plyne z faktu, že počet těchto funkcí je roven dimenzi celého prostoru, jejich počet je roven:

$$1 + m + \binom{m}{2} + \ldots + \binom{m}{m},$$

a to je číslo $n = 2^m$ (podle binomické věty, aplikované na $(1 + 1)^m$).

5.2 Reedovy-Mullerovy kódy

Reedovy-Mullerovy kódy (čti rídovy-málerovy) jsou speciální binární kódy délky $n = 2^m$. Kódová slova jsou booleovskými polynomy m proměnných. Reedovým-Mullerovým kódem (R-M kódem) stupně r a délky 2^m se nazývá množina $R(r, m)$ všech booleovských polynomů m proměnných stupně nejvýše r. R-M kódy je možno vytvářet postupně modifikovanými součty podle relace:

Reedovy-Mullerovy kódy

$$R(r, m + 1) = \mathrm{R}(r, m) + \mathrm{R}(r - 1, m), \text{ kde } m > r > 0. \qquad (58)$$

Příklad 5.2: Všechny R-M kódy délky 4 (m = 2)

0	0000	$R(-1, 2)$
1	1111	$R(0, 2)$
x_1	0101	$R(1, 2)$
x_2	0011	
$x_1 + x_2$	0110	
$1 + x_1$	1010	
$1 + x_2$	1100	
$1 + x_1 + x_2$	1001	
x_1x_2	0001	$R(2, 2)$
$1 + x_1x_2$	1110	
$x_1 + x_1x_2$	0100	
$x_2 + x_1x_2$	0010	
$x_1 + x_2 + x_1x_2$	0111	
$1 + x_1 + x_1x_2$	1011	
$1 + x_2 + x_1x_2$	1101	
$1 + x_1 + x_2 + x_1x_2$	1000	

opakovací kód

Některé R-M kódy jsou nám již známy. Například $R(-1, 2)$ je nulový vektor, $R(0, 2)$ je opakovací (4, 1)-kód. Kód $R(1, 2)$ je (4, 3)-kód celkové kontroly parity. Nakonec R-M kód $R(2, 2) = Z_2^4$

R-M kódy jsou zobecněním lineárních binárních kódů. Jejich hlavní přínos je v objevování vícenásobných chyb I přesto, že jsou k dispozici lepší kódy, používají se zejména tam, kde nejsou velké nároky na rychlost.

Kód $R(r, m)$ je lineární, protože součet dvou polynomů stupně $\leq r$ je také polynom stupně $\leq r$. Řádky generující matice tvoří všechny součiny $1, x_i, x_{i_1}x_{i_2}, ..., x_{i_1}x_{i_2}...x_{i_s}$ pro $s \leq r$.
Například kód $R(2, 3)$ má tuto generující matici (tvořenou booleovskými polynomy stupně ≤ 2):

$$\mathbf{G} = \begin{array}{|l|l|} \hline 11111111 & 1 \\ \hline 01010101 & x_1 \\ 00110011 & x_2 \\ 00001111 & x_3 \\ \hline 00010001 & x_1x_2 \\ 00000101 & x_1x_3 \\ 00000011 & x_2x_3 \\ \hline \end{array}$$

Všimněme si, že první řádek tvoří generující matici kódu $R(0, 3)$, první čtyři řádky tvoří generující matici kódu $R(1, 3)$,

5.3 Kódování R-M kódů

kódování R-M kódů

Při kódování v $R(r, m)$ jsou informační znaky hodnoty q_1 (= 0, 1) pro všechna čísla i = 0, 1, …, 2^m-1, jejichž binární rozvoj má nejvýše m jedniček. Pak vyšleme booleovský polynom $f = \sum_{i=0}^{2^m-1} q_i x_1^{i_1} x_2^{i_2} ... x_m^{i_m}$, kde klademe q_i = 0, jakmile má i v binárním rozvoji váhu $\geq m$.

Počet znaků kódu $R(r, m)$ je číslo:

$$k = \binom{m}{0} + \binom{m}{1} + \binom{m}{2} + ... + \binom{m}{m}. \qquad (59)$$

.
Což plyne ze skutečnosti, že počet všech součinů $x_{i_1}x_{i_2}...x_{i_s}$ (pro $i_1, i_2, \ldots, i_s$ navzájem různá) je $\binom{m}{s}$ pro s = 0, 1, …, r. Odtud vyplývá:

$$R(m, m) = Z_2^n \text{ pro } n = 2^m = \sum_{i=0}^{m} \binom{m}{i}, \qquad (60)$$

$R(m-1, m)$ je $(2^m, 2^m - 1)$-kód, tedy kód celkové kontroly parity,
$R(m-2, m)$ je $(2^m, 2^m - m - 1)$-kód, tedy rozšířený Hammingův kód,
$R(1, m)$ je $(2^m, m + 1)$-kód, duální k R$(m-2, m)$,
$R(0, m)$ je $(2^m, 1)$-kód, je opakovací kód,
$R(-1, m) = \{\mathbf{0}\}$.

5.4 Dekódování R-M kódů

Největší význam R-M kódů je v jejich jednoduché a snadno algoritmizovatelné metodě dekódování. Je založena na většinovém přístupu. Pro hledanou hodnotu najdeme soustavu rovnic a rozhodneme buď pro hodnotu 0 nebo 1 tak, aby platila většina těchto rovnic.

dekódování R-M kódů

Opakovací kódy *R*(0, *m*). dekódování provádíme metodou „hlasování“. Přijaté slovo $\mathbf{w} = w_0 w_1 \ldots w_{n-1}$, pro $n = 2^m$. Hledáme složku v_0 vyslaného vektoru tak, že položíme:

kód R(0, m)

$v_0 = w_0$,
$v_0 = w_1$,
…
$v_0 = w_{n-1}$,

Jestliže platí většina těchto rovnic pro $v_0 = 0$, rozhodneme, že skutečně platí $v_0 = 0$. Podobně pro $v_0 = 1$. V nerozhodném případě buď necháme v_0 nedefinováno, nebo (obvyklejší postup, který však nemění počet opravených chyb) položíme $v_0 = w_0$. Nakonec položíme $\mathbf{v} = v_0 v_0 \ldots v_0$.

Pokud nastalo méně než $n/2$ chyb, je většina z uvedených rovnic platná a vyslané slovo je skutečně **v**. Tím byl naplněn cíl, minimální vzdálenost kódu je n a jsme schopni opravit $n/2 - 1$ chyb.

Kódy prvního řádu *R*(1, *m*). Tyto kódy mají minimální Hammingovu vzdálenost $d = 2^m - 1$. Pro každý koeficient vyslaného slova v najdeme $2^m - 1$ rovnic a rozhodneme se pro tu hodnotu, která vyhovuje více z nich.
Kódové slovo je polynom nejvýše prvního stupně, tj.

kód R(1, m)

$$\mathbf{v} = q_0 1 + q_1 x_1 + q_2 x_2 + q_4 x_3 + q_8 x_4 + \ldots + q_{2^m - 1} x_m .$$

To znamená, že všechny koeficienty q_i, kde i není 0 nebo mocnina dvou, jsou nulové. Postupně určujeme koeficienty $q_1, q_2, \ldots, q_{2^m - 1}$ pomocí složek přijatého slova **w**.

Pro koeficient q_1 chceme sestavit $2^m - 1$ rovnic. Protože $M(1) = \{0, 1\}$ je jedna rovnice:
$q_1 = w_0 + w_1$.
Další rovnici najdeme, když si uvědomíme, že $q_3 = 0$, protože náš polynom je prvního stupně a neobsahuje tedy člen $x_1 x_2$. To znamená že $0 = w_0 + w_1 + w_2 + w_3 = q_1 + w_2 + w_3$, a další rovnice je:
$q_1 = w_2 + w_3$.

Podobně $q_5 = 0 = w_0 + w_1 + w_4 + w_5 = q_1 + w_4 + w_5$ a tedy platí:
$q_1 = w_4 + w_5$.

Zjišťujeme, že pro každé sudé číslo s = 0, …, $2^m - 1$ platí
$q_1 = w_s + w_{s+1}$.

Počet všech sudých čísel $s \leq 2^m - 1$ je 2^{m-1}, takže snadno najdeme 2^{m-1} rovnic. Protože pravé strany dvou různých rovnic neobsahují společné sčítance, žádná chyba neovlivní více než jednu rovnici. O hodnotě koeficientu q_1 rozhodneme hlasováním a v nerozhodném případě se řídíme první rovnicí. Pokud nastalo méně než polovina z 2^{m-1} chyb, určili jsme koeficient q_1 správně.

Obdobně postupujeme u všech koeficientů q_i, kde $i = 2^t$ (t = 0, 1, …, $m - 1$) a tedy $M_i = \{0, i\}$. Jedna rovnice je:
$q_i = w_0 + w_i$.

Další rovnici získáme tak, že si uvědomíme, že koeficient u členu $x_i x_u$ je nulový pro každé $u \neq t$. To znamená, že pro číslo $s = 2^u$ platí:
$q_{i+s} = 0$. (Binární rozvoj čísla $i + s = 2t + 2u$ má jen dvě jedničky na místech t a u). Platí tedy: $0 = w_0 + w_i + w_s + w_{i+s}$, a odtud získáme:
$q_i = w_s + w_{i+s}$.

Stejnou úvahu můžeme provést pro každé číslo $s \neq 0$ takové, že nemá jedničku na místě t v binárním rozvoji. Protože koeficient q_{i+s} patří ke členu stupně ≥ 2, je $q_{i+s} = 0$, a platí $q_i = w_s + w_{i+s}$. Jak vybrat vhodná čísla s? Číslo $2^m - 1$ má binární rozvoj 11…1, a tedy číslo $2^m - 1 - i$ má jedničky všude kromě místa t. Uvažujeme tedy všechna čísla $M(2^m - i - 1) = M(n - i - 1)$. Dostáváme rovnice:
$q_i = w_s + w_{i+s}$ (pro $s \in M(n - i - 1)$).

O hodnotě koeficientu q_i rozhodne většina rovnic, v nerozhodném případě první rovnice.

Jakmile jsme určili všechny koeficienty $q_1, q_2, ..., q_{2^m-1}$, odečteme příslušné slovo od slova **w**:
$\mathbf{w}' = \mathbf{w} - (q_1 x_1 + q_2 x_2 + ... + q_{2^m-1} x_{2^m-1})$.

Tím získáme polynom **w**' stupně 0 (pokud nedošlo k chybě), tedy slovo opakovacího kódu. To opět dekódujeme hlasováním, abychom určili q_0:
$q_0 = w_0' = w_1' = ... = w_{n-1}'$
Na závěr položíme:
$\mathbf{v} = q_0 1 + q_1 x_1 + ... + q_{2^m-1} x_m$

Příklad 5.3: Používáme kód $R(1, 3)$ a chceme dekódovat slovo **w** = 11101010. Platí:

$\mathbf{v} = q_0 1 + q_1 x_1 + q_2 x_2 + q_4 x_3$

K určení koeficientu q_1 máme rovnice:

$q_1 = w_0 + w_1 \quad (s = 0)$
$q_1 = w_2 + w_3 \quad (s = 2)$
$q_1 = w_4 + w_5 \quad (s = 4)$
$q_1 = w_6 + w_7 \quad (s = 6)$

Kromě první rovnice všechny platí pro $q_1 = 1$.

K určení koeficientu q_2 máme rovnice:

$q_2 = w_0 + w_2 \quad (s = 0)$
$q_2 = w_1 + w_3 \quad (s = 1)$
$q_2 = w_4 + w_6 \quad (s = 4)$
$q_2 = w_5 + w_7 \quad (s = 5)$

Kromě druhé rovnice všechny platí pro $q_2 = 0$.

K určení koeficientu q_4 máme rovnice:

$q_4 = w_0 + w_4 \quad (s = 0)$
$q_4 = w_1 + w_5 \quad (s = 1)$
$q_4 = w_2 + w_6 \quad (s = 2)$
$q_4 = w_3 + w_7 \quad (s = 3)$

Kromě druhé rovnice všechny platí pro $q_4 = 0$.

Zbývá určit koeficient q_0. Od přijatého slova odečteme $q_1 x_1$ (protože $q_2 = q_4 = 0$), pro x_1 = 01010101 vypočítáme:
$\mathbf{w}' = \mathbf{w} - q_1 x_1$ = 11101010 – 0 1010101 = 10111111
odtud hlasováním dostáváme $q_0 = 1$.

Výsledný vektor je
$\mathbf{v} = q_0 + q_1 x_1$ = 11111111 + 01010101 = 10101010. Pokud tedy nedošlo k více než jedné chybě, zjistili jsme, že při přijetí slova **w** = 11101010 bylo vysláno slovo **v** = 10101010.

kód R(r, m)

Obecné kódy *R*(*r*, *m*). Každé kódové slovo **v** je booleovským polynomem

$$\mathbf{v} = \sum_{i=0}^{n-1} q_i x_m^{i_m} \ldots x_2^{i_2} x_1^{i_1} \ .$$

Pro dekódování provádíme postupně jednotlivé iterační kroky. Pro hledanou soustavu rovnic se rozhodneme buď pro 0 nebo 1 tak, aby většina rovnic platila.

1. Přijmeme slovo **w** a určíme koeficient s indexem váhy r:

   Hledáme slovo $\mathbf{v} = \sum_{i=0}^{n-1} q_i x_m^{i_m} \ldots x_2^{i_2} x_1^{i_1}$

   Z přijatého slova $\mathbf{w} = w_0 w_1 \ldots w_{n-1}$, pro $n = 2^m$. Je-li většina $q_i = 1$ bude $q_i = 1$, pokud je většina $q_i = 0$ bude $q_i = 0$. Pokud je situace nerozhodná, bude q_i nabývat minulé hodnoty.

2. Následuje rekurzívní krok, v němž stejným postupem určujeme

$$\mathbf{w}' = \mathbf{w} - \sum_{\|j\|=r} q_j x_m^{j_m} \ldots x_2^{j_2} x_1^{j_1}$$

Předchozí postup aplikujeme na slovo **w**'. Opakujeme dokud nezískáme hodnotu q_i.

3. Opravíme *i*-tý bit slova.

Tato metoda umožňuje opravování chyb v počtu $2^{m-r-1} - 1$. To je zajištěno Hammingovou vzdáleností 2^{m-r}. Například kód $R(1, 5)$ je označován také jako (32, 6)-kód, který opravuje $2^{5-1-1} - 1 = 7$ chyb.

Reedovy-Mullerovy kódy byly objeveny v roce 1954. První významné použití kódu $R(1, 4)$ tedy (16, 5)-kódu a $R(1, 5)$ tedy (32, 6)-kódu z roku 1969 souvisí se zabezpečením rádiového přenosu fotografií sondy Mariner 9 z povrchu Marsu.

Kontrolní otázky:

14. Co je booleovský polynom?.
15. Jak vypadá booleovská funkce?
16. Jak jsou definovány Reedovy-Mullerovy kódy?
17. Jak probíhá kódování R-M kódu?
18. Jak probíhá dekódování R-M kódu?

Shrnutí obsahu kapitoly

V této kapitole jsme se seznámili se zajímavou třídou Reedových-Mullerových kódů, první objevenou skupinou kódů schopných opravovat volitelný počet chyb. Současně jsme se seznámili s booleovskými polynomy a booleovskými funkcemi, které můžeme využít také v oblasti logických systémů. Umíme sestavit R-M kód i popsat algoritmus jeho dekódování včetně opravy chyb.

6 Cyklické kódy

Cíl:

Cílem celé kapitoly je představit významnou třídu lineárních kódů – kódy cyklickými, nazývaných také jako kódy polynomické, protože pro jejich definici s výhodou využíváme polynomy. jejich významnou výhodou je schopnost odhalovat kromě náhodných chyb také chyby shlukové.

Po jejím prostudování byste měli být schopni:

- Pracovat s polynomy, zejména násobit je a dělit.
- Definovat základní principy cyklických kódů, generující a kontrolní polynom.
- Sestavit cyklický kód.
- Zkontrolovat správnost přeneseného slova zabezpečeného cyklickým kódem.
- Realizovat cyklický kód také v hardwarové podobě.

Klíčová slova této kapitoly:

Cyklický kód, polynomický kód, okruh polynomů, báze kódu, generující polynom, generující matice, kontrolní polynom, kontrolní matice, systematický kód, systematické kódování, shlukové chyby, paměťový kanál, CRC.

Doba potřebná ke studiu: 5 hodin

Průvodce studiem

Studium této kapitoly je nutné k pochopení principů konstrukce cyklických kódů, které jsou významnou podmnožinou lineárních kódů. Použití booleovských funkcí a booleovských polynomů je využitelné i při optimalizaci logických obvodů a systémů logického řízení.
Na studium této části si vyhraďte pět hodin. Zkuste si samostatně vyřešit také všechny příklady tvorby cyklických kódů a jejich použití.

cyklický kód

polynomický kód

Cyklické kódy jsou rozsáhlou třídou zabezpečovacích kódů. Podle způsobu manipulace s kódovým slovem se také označují jako ***polynomické kódy***.
Kódová slova totiž můžeme chápat jako zkrácený zápis polynomu stupně *n*:

$$a_0 + a_1 z + a_2 z^2 + ... a_{n-1} z^{n-1} \rightsquigarrow \quad a_0 a_1 a_2 a_3 ... a_{n-1}. \tag{61}$$

Lineární kód se pak nazývá cyklický, jestliže s každým kódovým slovem $a_0 a_1 a_2 a_3 ... a_{n-1}$ obsahuje také slovo $a_{n-1} a_0 a_1 ... a_{n-2}$, které vzniklo cyklickým posunem jednotlivých bitů. Při práci s polynomy přitom odpovídá cyklický posun násobení proměnnou *z*. Přesun koeficientu u nejvyšší mocniny na začátek kódového slova vyřešíme zavedením:

$$z^n = 1, z^{n+1} = z, z^{n+2} = z^2, ... \tag{62}$$

K definici cyklických kódů jsou zapotřebí znalosti z algebry polynomů a faktorových okruhů. Jejich objasnění přesahuje rámec této publikace, blíže je možno se s nimi seznámit např. v [1]. Pro nás bude nejvýznamnější operací násobení a dělení polynomů. Dělení polynomu a(*z*) polynomem b(*z*) chápeme jako určení podílu q(*z*) a zbytku r(*z*). Přitom platí:

$$\mathrm{a}(z) = \mathrm{q}(z) \cdot \mathrm{b}(z) + \mathrm{r}(z) \qquad \text{a} \qquad \deg\ \mathrm{r}(z) < \deg\ \mathrm{b}(z) \tag{63}$$

kde je deg – stupeň polynomu – nejvyšší číslo $k = \deg \mathrm{a}(z)$, takové, že $a_k \neq 0$; stupeň nulového polynomu přitom předpokládáme roven –1.

okruh polynomů

Pro definici cyklických kódů je významný pojem ***okruh polynomů modulo*** q(*z*):

$$\mathrm{T} / \mathrm{q}(z), \tag{64}$$

kde je T – těleso vzniklé z abecedy T, u binárních kódů pracujeme s tělesem $Z_2 = \{0, 1\}$,
q(*z*) – polynom proměnné *z* nad tělesem T, koeficienty polynomu jsou prvky tělesa T.

Základní operace v okruhu polynomů modulo q(*z*) jsou pak sčítání, které se neliší od běžného sčítání polynomů:
a(*z*) + b(*z*).

Násobení ⊗ je však definováno jako zbytek po dělení polynomu a(*z*)·b(*z*) polynomem q(*z*)
a(*z*) ⊗ b(*z*).

Pokud zvolíme $\mathrm{q}(z) = z^n - 1$ vznikne okruh polynomů, ve kterém platí $z^n = 1$, což je požadavek pro cyklický kód (62). Polynomy patřící do tohoto okruhu pak definují jednotlivá kódová slova. Ukažme si základní operace s polynomy v okruhu polynomů modulo q(*z*) na příkladu okruhu $Z_2 / (z^2 - 1)$. Jeho prvky jsou polynomy stupně ≤ 1: 0, 1, *z*, *z* + 1.

Pro sčítání platí tabulka:

+	0	1	z	z+1
0	0	1	z	z+1
1	1	0	z+1	z
z	z	z+1	0	1
z+1	z+1	z	1	0

Násobení je určeno jako zbytek po dělení $z^2 - 1$, např.: $z \otimes z = z^2 - \left(z^2 - 1\right) = 1$

Pro násobení platí tabulka:

$\otimes$	0	1	z	z+1
0	0	0	0	0
1	0	1	z	z+1
z	0	z	1	z+1
z+1	0	z+1	z+1	0

Všimněte si zejména násobení polynomem z, které odpovídá cyklickému posunu.

generující polynom

Cyklický kód můžeme kromě generující matice (jako každý jiný lineární kód) definovat pomocí ***generujícího polynomu*** g(z). Generující polynom cyklického (n, k)-kódu je polynom stupně $n - k$ v tomto kódu, který je dělitelem polynomu ($z^n - 1$).

generující matice

Generující matice cyklického kódu vznikne cyklickým posunem koeficientů generujícího polynomu:

k-1

$$\mathbf{G} = \begin{bmatrix} g_0 & g_1 & g_2 & \dots & g_{n-k} & 0 & 0 & \dots & 0 \\ 0 & g_0 & g_1 & \dots & g_{n-k-1} & g_{n-k} & 0 & \dots & 0 \\ . & & & & & & & & \\ . & & & & & & & & \\ . & & & & & & & & \\ 0 & 0 & \dots & 0 & g_0 & g_1 & \dots & \dots & g_{n-k} \end{bmatrix}. \qquad (65)$$

k-1

Její řádky tvoří polynomy

$$\mathrm{g}(z)$$
$$z \cdot \mathrm{g}(z)$$
.
.
.
$$z^{k-1} \cdot \mathrm{g}(z)$$

báze kódu

Z definice generující matice víme, že tyto polynomy tvoří ***bázi*** kódu. Pro generující polynom g(z) (n, k)-kódu K tedy platí:

1. Stupeň polynomu g(z) je $n - k$,
2. Kód K sestává ze všech násobků polynomu g(z) v prostoru T^n, tedy:
$$\mathrm{K} = \{\mathrm{q}(z)\mathrm{g}(z) | \mathrm{q}(z) \in \mathrm{T}^n\},$$
3. Polynom g(z), z.g(z), ..., z^{k-1}.g(z) tvoří bázi kódu K,
4. Polynom g(z) dělí polynom $z^n - 1$ beze zbytku.

kontrolní polynom

U cyklických kódů můžeme kontrolní matici nahradit ***kontrolním polynomem***:

$$\mathrm{h}(z) = (z^n - 1) : \mathrm{g}(z) \qquad (66)$$

kontrolní matice

Kontrolní matici cyklického (n, k)-kódu získáme cyklickými posuvy koeficientů kontrolního polynomu čteného od nejvyšší mocniny:

$$\mathbf{H} = \begin{bmatrix} \overbrace{0 \quad 0 \quad . \quad . \quad . \quad 0 \quad 0}^{n-k+1} & h_k & h_{k-1} & . & . & . & h_1 & h_0 \\ 0 \quad 0 \quad . \quad . \quad . \quad 0 & h_k & h_{k-1} & . & . & . & h_1 & h_0 & 0 \\ . \\ . \\ . \\ h_k & h_{k-1} & . & . & . & h_1 & h_0 & . & . & . & 0 \end{bmatrix} . \qquad (67)$$

Kontrolní matice má $n - k$ řádků. Protože $h_k \neq 0$, jsou řádky lineárně nezávislé. Pro každý polynom v(z), pro který platí:

$$\mathrm{v}(z) \cdot \mathrm{h}(z) = 0, \qquad (68)$$

splňuje kontrolní matice podmínku

$$\mathbf{H} \cdot \mathbf{v}^{\mathrm{T}} = \mathbf{0}^{\mathrm{T}} \cdot \qquad (69)$$

Příklad 6.1: Kód celkové kontroly parity (sudá parita) délky 4 bity, tedy (4, 3)-kód je lineární kód, který je také cyklickým kódem. Skládá se z těchto polynomů:

$$0, 1+z, 1+z^2, 1+z^3, z+z^2, z+z^3, z^2+z^3, 1+z+z^2+z^3$$

Nenulový kódový polynom nejnižšího stupně je polynom $1 + z$, ostatní kódové polynomy jsou všechny násobky polynomu $1 + z$ v okruhu $Z_2/(z^4 - 1)$. Přitom platí:

$$\left(z^4-1\right)=\left(z^3-z^2+z-1\right)\left(z+1\right). \tag{70}$$

Jeho generující matici podle vztahu (65) určíme jako:

$$\mathbf{G}=\begin{bmatrix}1 & 1 & 0 & 0\\ 0 & 1 & 1 & 0\\ 0 & 0 & 1 & 1\end{bmatrix}. \tag{71}$$

Jeho kontrolní polynom vyplývá ze vztahů (66) a (70) (je třeba si uvědomit, že v binární aritmetice platí $-z = z$):

$$\mathrm{h}(z)=z^3+z^2+z+1. \tag{72}$$

Z něj určíme kontrolní matici

$$\mathbf{H}=\begin{bmatrix}1 & 1 & 1 & 1\end{bmatrix}, \tag{73}$$

což odpovídá našim dosavadním poznatkům o tomto kódu.

Příklad 6.2: (4, 2)-kód s generující maticí:

$$\mathbf{G}=\begin{bmatrix}1 & 1 & 1 & 1\\ 0 & 1 & 0 & 1\end{bmatrix} \tag{74}$$

má kódová slova 0000, 1111, 0101, 1010. Je to tedy cyklický kód. Polynom stupně $n - k = 4 - 2 = 2$ je 1010, tedy $1 + z^2$. Ten však dává jinou generující matici:

$$\mathbf{G}'=\begin{bmatrix}1 & 0 & 1 & 0\\ 0 & 1 & 0 & 1\end{bmatrix}. \tag{75}$$

Kontrolní polynom je $\mathrm{h}(z)=1+z^2$, neboť platí:

$$(z^4-1)=(z^2+1)(z^2-1). \tag{76}$$

Kontrolní matice tohoto kódu je pak:

$$\mathbf{H}=\begin{bmatrix}0 & 1 & 0 & 1\\ 1 & 0 & 1 & 0\end{bmatrix}. \tag{77}$$

Generující matice $\mathbf{G}'$ (75) odpovídá generující matici systematického kódu, můžeme ji tedy zapsat jako $\mathbf{G}=\left[\mathbf{E}\middle|\mathbf{B}\right]$ a kontrolní matice je $\mathbf{H}=\left[-\mathbf{B}^T\middle|\mathbf{E}'\right]$. Takto získáme opět jinou kontrolní matici:

$$\mathbf{H}'=\begin{bmatrix}1 & 0 & 1 & 0\\ 0 & 1 & 0 & 1\end{bmatrix}. \tag{78}$$

Příklad 6.3: Z dříve uvedených poznatků vyplývá, že každý cyklický kód rozkládá polynom $z^n - 1$ na součin g(*z*).h(*z*). To platí i naopak. Popišme nyní všechny binární cyklické kódy délky 5. Platí:

$$z^5 - 1 = (z+1)(z^4 + z^3 + z^2 + z + 1), \tag{79}$$

odtud dostáváme dva kódy (netriviální kódy):

1. kód celkové kontroly parity $g(z) = z + 1$, $h(z) = z^4 + z^3 + z^2 + z + 1$,
2. opakovací kód $g(z) = z^4 + z^3 + z^2 + z + 1$, $h(z) = z + 1$.

Příklad 6.4: Hammingův (7, 4)-kód není cyklický. Např. cyklickým posunem slova 1101001 (první řádek generující matice) dostaneme nekódové slovo 1110100 (jeho syndrom je 101). Protože sloupce kontrolní matice Hammingova kódu můžeme psát v libovolném pořadí, uspořádáme je tak, aby vznikl cyklický kód.

$$\mathbf{H} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}. \tag{80}$$

Sloupce tvoří všechna slova délky 3 kromě 000. Místo slov budeme psát polynomy stupně nejvýše 2 v pomocné proměnné *x* (její označení je voleno z důvodu odlišení od proměnné z). Chceme najít pořadí, v jakém zapsat všechny nenulové polynomy $a + bx + cx^2$ jako sloupce

$$\begin{bmatrix} c \\ b \\ a \end{bmatrix}$$

matice **H** tak, aby vznikl cyklický kód. K tomu použijeme okruh $Z_2 / q(x)$, kde je q(*x*) polynom třetího stupně, takže prvky okruhu jsou právě naše polynomy. Jako vhodná volba se ukazuje:

$q(x) = x^3 + x + 1$,

neboť platí $x^3 + x + 1 = 0$, takže mocninami x^i vyjádříme naše polynomy:

$$\begin{aligned} x^0 &= 1 \\ x^1 &= x \\ x^2 &= x^2 \\ x^3 &= x + 1 \\ x^4 &= x^2 + x \\ x^5 &= x^2 + x + 1 \\ x^6 &= x^2 + 1 \end{aligned} \tag{81}$$

Kontrolní matici **H** nyní uspořádáme podle těchto mocnin:

$$\mathbf{H} = \begin{bmatrix} 1 & x & x^2 & x^3 & x^4 & x^5 & x^6 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}. \tag{82}$$

Kód s touto kontrolní maticí sestává ze všech polynomů

$$\mathrm{v}(z) = v_0 + v_1 z + \ldots + v_6 z^6 , \tag{83}$$

pro které platí:

$$\mathbf{H}\begin{bmatrix} v_0 \\ v_1 \\ . \\ . \\ . \\ v_6 \end{bmatrix} = \begin{bmatrix} 1 & x & x^2 & . & . & . & x^6 \end{bmatrix}\begin{bmatrix} v_0 \\ v_1 \\ . \\ . \\ . \\ v_6 \end{bmatrix} = v_0 + v_1 x + v_2 x^2 + \ldots + v_6 x^6 = 0 , \tag{84a}$$

neboli

$$\mathrm{v}(x) = 0 \text{ v okruhu } Z_2 /(x^3 + x + 1) , \tag{84b}$$

a tento kód je cyklický, přitom má kontrolní matice jako sloupce všechna nenulová slova délky 3, takže je to Hammingův (7, 4)-kód. Generující polynom je stupně 7 – 4 = 3 a je to jediný takový polynom

$$\mathrm{g}(z) = z^3 + z + 1 . \tag{85}$$

Generující matice je tedy:

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} . \tag{86}$$

Kontrolní polynom h(z) určíme ze vztahu (66):

$$\mathrm{h}(z) = (z^7 - 1) : (z^3 + z + 1) = z^4 + z^2 + z + 1 . \tag{87}$$

Ten určuje kontrolní matici:

$$\mathbf{H}' = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix} , \tag{88}$$

ta vznikne z kontrolní matice (82) přičtením prvního řádku k poslednímu.

6.1 Realizace cyklických kódů

systematický kód

Dosud nebyla řešena otázka realizace cyklických kódů. Jak bylo uvedeno, cyklický kód je kódem lineárním, jako takový je ekvivalentní se ***systematickým kódem***. U systematického kódu můžeme oddělit informační bity od kontrolních bitů. Tím se velmi usnadňuje dekódování.

systematické kódování

Pro cyklické kódy existuje velmi jednoduché ***systematické kódování***. Každé kódové slovo čteme pozpátku (neboli polynomy zapisujeme od nejvyšší mocniny, tedy opačně než jsme je zapisovali dosud). Z informačních bitů $u_0\ u_1\ ...\ u_{k-1}$ vytvoříme polynom:

$$\mathrm{u}(z) = u_0 z^{n-1} + u_1 z^{n-2} + ... + u_{k-1} z^{n-k} . \qquad (89)$$

Tento polynom dělíme generujícím polynomem g(z):

$$\mathrm{u}(z) = \mathrm{q}(z)\mathrm{g}(z) + \mathrm{r}(z) \text{, kde je deg } \mathrm{r}(z) < n-k . \qquad (90)$$

Odečtením zbytku vznikne kódové slovo:

$$\mathrm{q}(z)\mathrm{g}(z) = \mathrm{u}(z) - \mathrm{r}(z) , \qquad (91)$$

které je nutně slovem cyklického kódu. Protože v binární aritmetice platí $1+1=0$, polynom u(z) obsahuje jen koeficienty u mocniny $n - k$ nebo vyšší, zatímco zbytek koeficienty nižší, pak při označení

$$\mathrm{r}(z) = r_k z^{n-k-1} + r_{k+1} z^{n-k-2} + ... + r_{n-1} z + r_n , \qquad (92)$$

vyšleme kódové slovo:

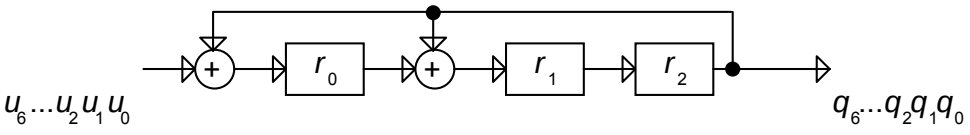
$$u_0 u_1 ... u_{k-1} r_k r_{k+1} ... r_n . \qquad (93)$$

Výhodou uvedeného postupu je skutečnost, že u binárních kódů, kde platí r(z) + r(z) = 0 vysíláme kódové slovo u(z) + r(z), které je bezezbytku dělitelné generujícím polynomem. Kontrolu správnosti přeneseného kódového slova tedy zjistíme jeho dělením generujícím polynomem.

Např. v rozebíraném Hammingově (7, 4)-kódu zakódujeme informační bity 1010 ($u_0u_1u_2u_3$) tak, že polynom u(z) = $z^6 + z^4$ dělíme generujícím polynomem g(z) = $z^3 + z + 1$ a získáme zbytek r(z) = $z + 1$. Výsledné kódové slovo je u(z) – r(z) = $z^6 + z^4 + z + 1$. Při zápisu od nejvyšší mocniny tedy vyšleme 1010011.

Při kontrole přijatého kódového slova již postupujeme jako dříve. Přijaté kódové slovo je v(z) = $z^6 + z^4 + z + 1$. Dělením kontrolním polynomem h(z) = $z^4 + z^2 + z + 1$ zjistíme, že kódové slovo je dělitelné beze zbytku a tedy správné. Totéž zjistíme dělením generujícím polynomem. Obdobně pro informační bity 1001 získáme kódové slovo 1001110.

Vlastní operace dělení generujícím polynomem je snadno realizovatelná pomocí dvou binárních sčítaček a tří posuvných registrů (pro Hammingův (7, 4)-kód):



Obr. 6. Hardwarová realizace dělení generujícím polynomem g(z) = $z^3 + z + 1$

Do obvodu vstupují koeficienty polynomu u(z) = $u_0z^6 + u_1z^5 + ... + u_6$ a vystupují koeficienty podílu q(z) = $q_0z^6 + q_1z^5 + ... + q_6$. Po vystoupení posledního koeficientu zůstávají v registrech koeficienty zbytku r(z) = $r_2z^2 + r_1z + r_0$.

Např. při dělení u(z) = $z^6 + z^4$ (kódování informačních bitů 1010) bude obsah registrů postupně nabývat hodnot:

vstup		r_0	r_1	r_2	**výstup**	
u_0	1	1	0	0	0	q_0
u_1	0	0	1	0	0	q_1
u_2	1	1	0	1	0	q_2
u_3	0	1	0	0	1	q_3
u_4	0	0	1	0	0	q_4
u_5	0	0	0	1	0	q_5
u_6	0	1	1	0	1	q_6

Získali jsme tedy q(z) = $z^3 + 1$ a r(z) = $z + 1$.
V souladu se systematickým kódováním bude kódové slovo 1010011.
Obdobně získáme pro dělení u(z) = $z^6 + z^3$ (kódování informačních bitů 1001) q(z) = $z^3 + z$ a r(z) = $z^2 + z$, tedy příslušné kódové slovo bude 1001110.
Správnost činnosti můžeme snadno ověřit, pokud si uvědomíme, že posun registrů vpravo představuje násobení jejich obsahu proměnnou z. Zpětná vazba pak realizuje modulování generujícím polynomem. Pokud původní obsah registrů byl

$$r_2z^2 + r_1z + r_0, \tag{94}$$

pak posunem o jednu pozici doprava získáme polynom

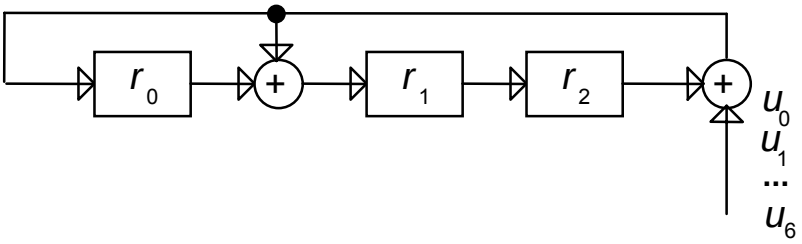
$$r_1z^2 + (r_2 + r_0)z + r_2. \tag{95}$$

Protože platí $r_2z^3 + r_2z^3 = 0$, můžeme tento vztah rozšířit a upravit do tvaru

$$z(r_2z^2 + r_1z + r_0) + r_2(z^3 + z + 1) = z \cdot r(z) \bmod g(z). \tag{96}$$

Jestliže tedy registry na počátku vynulujeme a koeficienty polynomu u(z) budou vstupovat počínaje koeficientem u nejvyšší mocniny, pak po vstoupení posledního koeficientu bude v registrech uloženo $(u_0z^{n-1} + u_1z^{n-2} + ... + u_kz^{n-k} + 0...0) \bmod g(z)$, tedy skutečně zbytek po dělení generujícím polynomem.
Pro vlastní realizaci kódování bude vhodné použitý obvod mírně upravit. Protože podíl q(z) je pro nás nepodstatný, zajímá nás pouze zbytek r(z), takže pokud přesuneme vstup informačních bitů na konec obvodu, získáme zbytek r(z) ihned po průchodu informačních bitů obvodem. Tím „ušetříme“ průchod posledních tří bitů (obecně $n - k$ bitů), které jsou vždy nulové, jak vyplývá z (89).

Upravený obvod pak vypadá následovně:



Obr. 7. Hardwarová realizace určení zbytku po dělení generujícím polynomem g(z)

Zbytek přitom je $\mathrm{r}(z) = r_2 z^2 + r_1 z + r_0$. Při kódování informačních bitů 1010 bude obsah registrů postupně nabývat hodnot:

vstup	r_0	r_1	r_2
start	0	0	0
1	1	1	0
0	0	1	1
1	0	0	1
0	1	1	0

Takže po průchodu pouze informačních bitů je v registrech hodnota zbytku r(z), který připojíme za informační bity a získáme kódové slovo 1010011. Při vysílání kódového slova budeme vysílat bity od nejvyšší mocniny z (od začátku slova), takže kontrola může probíhat souběžně s přijímáním znaků. Po přijetí informačních bitů a jejich vydělení g(z) známe zbytek, který srovnáme s přijatým zbytkem. Nebo necháme celé kódové slovo vydělit g(z), zbytek musí být 0.

6.2 Použití cyklických kódů

shlukové chyby

paměťový kanál

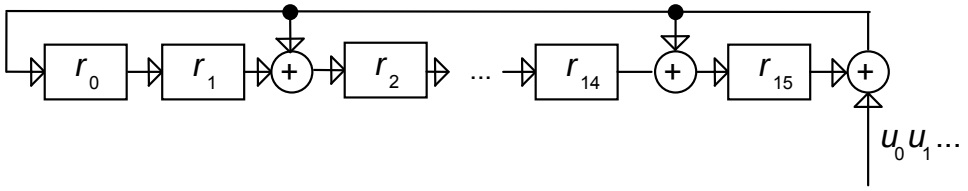
Používání cyklických kódů se v poslední době velmi rozšířilo. Důvodem je především schopnost cyklických kódů odhalovat ***shlukové chyby***. Zkoumáním informačních kanálů se totiž zjistilo, že se většinou chovají jako ***paměťové kanály*** a chyby mají tendenci vyskytovat se blízko sebe.

CRC

Nejznámější jsou bezesporu kódy ***CRC*** (Cyclic Redundancy Code) používané pro zjištění neporušenosti dat. Z každého bloku dat (obvykle násobky 256 Byte) je vypočten zbytek po dělení generujícím polynomem a uložen za tato data. Následující tabulka uvádí nejznámější kódy CRC [52]:

počet kontrolních bitů	označení	generující polynom	použití
8	LRCC–8	z^8+1	kontrolní Byte je součet datových Byte modulo 2
12	CRC–12	$z^{12}+z^{11}+z^3+z^2+z+1$	používá se pro šestibitové znaky
16	LCRC–16	$z^{16}+1$	kontrolní součet dvojic Byte (Word) modulo 2
16	CRC–16	$z^{16}+z^{15}+z^2+1$	binární synchronní protokol
16	CRC–16 reverzní	$z^{16}+z^{14}+z+1$	
16	SDLC	$z^{16}+z^{12}+z^5+1$	linkový protokol IBM, standard CCITT
16	SDLC reverzní	$z^{16}+z^{11}+z^4+1$	
32	CRC–32	$z^{32}+z^{26}+z^{23}+z^{22}+$ $+z^{16}+z^{12}+z^{11}+z^{10}+$ $+z^8+z^7+z^5+z^4+$ $+z^2+z+1$	Ethernet, HDLC, ZMODEM

Blíže si všimněme kódu CRC–16. Opět je možno ho realizovat pomocí binárních sčítaček a posuvných registrů. Využijeme dosavadních znalostí a necháme informační bity vstupovat na konec obvodu. Po vstupu posledního informačního bitu máme v registrech zbytek po jeho dělení generujícím polynomem.



Obr. 8. Hardwarová realizace určení zbytku po dělení generujícím polynomem g(z)

Po průchodu informačních bitů $u_0u_1 \ldots u_k$ je v registrech uložen zbytek $r_{15}r_{14} \ldots r_0$. Výsledné kódové slovo je pak tvořeno jejich spojením $u_0u_1 \ldots u_kr_{15}r_{14} \ldots r_0$. Např. slovo 10100001 dává zbytek 1000001111000101. Celé kódové slovo je pak 10100001100000111100010 1, což odpovídá zásadám systematického kódování informačních bitů, jak bylo popsáno.

Obrovskou výhodou CRC kódů je možnost jejich snadné softwarové realizace při zpracování celých Byte najednou (u CRC–16). Abychom si uvědomili možnosti softwarové realizace, vyjděme z následující úvahy [51].
Předpokládejme, že již máme zpracovánu část zabezpečovaného bloku dat d(z) a odpovídající stav registru r(z) = CRC(d(z)). Další přicházející Byte je $\mathrm{db}(z) = db_7 z^7 + ... + db_1 z + db_0$.

Rozšířeným datům $\mathrm{d}'(z) = z^8 \mathrm{d}(z) + \mathrm{db}(z)$ odpovídá CRC

$$\mathrm{r}'(z) = \left[z^{16} \mathrm{d}'(z)\right] \mathrm{mod}\, \mathrm{g}(z), \text{ což je po dosazení za } \mathrm{d}'(z)$$

$$\left[z^8 z^{16} \mathrm{d}(z) + z^{16} \mathrm{db}(z)\right] \mathrm{mod}\, \mathrm{g}(z). \qquad (97)$$

Přitom $z^{16} \mathrm{d}(z) \,\mathrm{mod}\, \mathrm{g}(z)$ je již vypočtené r(z), takže nový stav registrů je roven výrazu:

$$\mathrm{r}'(z) = \left[z^8 \mathrm{r}(z) + z^{16} \mathrm{db}(z)\right] \mathrm{mod}\, \mathrm{g}(z). \qquad (98)$$

Pokud vyjádříme jednotlivé bity registru r(z) – je šestnáctibitový a slova db(z) – je osmibitové a označíme $x_7 = db_7 + r_{15}, x_6 = db_6 + r_{14}, ..., x_0 = db_0 r_8$, pak rovnici (98) upravíme do tvaru:

$$\mathrm{r}'(z) = \left[z^{16}\left(db_7 z^7 + ... + db_0 z^0\right) + z^8\left(r_7 z^7 + ... + r_0 z^0\right) + z^{16}\left(r_{15} z^7 + ... + r_8 z^0\right)\right] \mathrm{mod}\, \mathrm{g}(z) =$$

$$= \left[z^8\left(r_7 z^7 + ... + r_0 z^0\right) + z^{16}\left(x_7 z^7 + ... + x_0 z^0\right)\right] \mathrm{mod}\, \mathrm{g}(z) =$$

$$= z^8 (r_7 z^7 + ... + r_0 z^0) + \mathrm{CRC}(\mathrm{x}(z)).$$

CRC rozšířených dat je tedy tvořeno dvěma složkami, přičemž druhá vznikne operací XOR dat db (8 bitů) a pravé poloviny slova r(z), tedy předchozího CRC. Výpočet pak probíhá tak, že si předem zpracujeme CRC všech 256 možných hodnot polynomu x(z) (8 bitů). Registr CRC vynulujeme a každý nový Byte db „přixorujeme“ k pravé polovině starého CRC, tím vypočteme x(z), vyhledáme v tabulce CRC(x(z)) a „přixorujeme“ ho ke staré hodnotě CRC posunuté o 8 bitů doprava. Tím získáme nové CRC. Podrobný popis softwarové realizace se nachází v [51, 52, 83].

Kontrolní otázky:

19. Jak jsou definovány cyklické kódy?.
20. Jak určíme generující a kontrolní polynom kódu?
21. Jak z generujícího polynomu určíme generující matici kódu?
22. Jak z kontrolního polynomu určíme kontrolní matici kódu?
23. Jak realizujeme systematické kódování cyklického kódu?
24. Informační slovo (1001) je třeba zabezpečit pomocí cyklického (7, 4)-kódu s generujícím polynomem g(z) = $z^3 + z + 1$. Ukažte postup systematického kódování.

Shrnutí obsahu kapitoly

V této kapitole jsme se seznámili s významnou třídou lineárních kódů – cyklickými kódy. K jejich definici s výhodou používáme polynomy a operace s nimi, zejména dělení polynomů. Umíme sestavit generující a kontrolní polynom kódu, zakódovat i dekódovat informační slovo kódu. Známe také významnou skupinu CRC kódů, používaných pro odhalení shlukových chyb při zabezpečení dat. Seznámili jsme se také s hardwarovou realizací cyklických kódů.

7 BCH-kódy

Cíl:

Cílem celé kapitoly je představit významnou třídu kódů BCH-kódů. Patří mezi lineární kódy a jsou schopny opravit dvojnásobné i vícenásobné chyby, to je činí nejdůležitější třídou cyklických kódů. Pro práci s BCH-kódy využíváme operace s tělesy polynomů, zejména na Galoisově tělese.

Po jejím prostudování byste měli být schopni:

- Pracovat s aritmetickými tělesy.
- Definovat základní typy BCH-kódů a jejich možnosti.
- Sestavit BCH-kód.

Klíčová slova této kapitoly:

BCH-kód, cyklický Hammingův kód, těleso, algebraické těleso, konečné těleso, těleso polynomů, Galoisovo těleso, primitivní prvek, nerozložitelný polynom, generující kořen, nejmenší společný násobek polynomů, maticová metoda dekódování, lokátor chyb.

Doba potřebná ke studiu: 4 hodiny

Průvodce studiem

Studium této kapitoly je nutné k pochopení principů konstrukce BCH-kódů. Operace s tělesy polynomů, zejména Galoisovými tělesy umožňují sestavit BCH-kód požadovaných vlastností .
Na studium této části si vyhraďte čtyři hodiny. Zkuste si samostatně vyřešit příklady tvorby BCH-kódů a jejich použití.

BCH-kód

BCH-kódy jsou schopny opravit dvojnásobné i vícenásobné chyby, to je činí nejdůležitější třídou cyklických kódů. BCH-kódy popsali na konci padesátých let Raj Chandra Bose, Dwijendra Kumar Ray-Chaudhuri a Alexis Hocquenghem, z iniciál jejich příjmení vzniklo později označení BCH-kódy. Asi největší význam mají pro opravu dvojnásobných chyb.

cyklický Hammingův kód

Pro opravu jednoduchých chyb slouží ***cyklické Hammingovy kódy***. Pro vyjádření schopnosti opravovat jednoduché chyby můžeme použít popis kódu pomocí generujících kořenů, jak bylo ukázáno na příkladu 4 v kapitole 4.4. To umožňuje do definice zahrnout také cyklické kódy pro opravu vícenásobných chyb. Například Hammingův (15, 11)-kód je popsán svojí generující maticí:

$$\mathbf{H} = \begin{bmatrix} 1 & \alpha & \alpha^2 & \ldots & \alpha^{14} \end{bmatrix}, \tag{99}$$

Galoisovo těleso

primitivní prvek

kde je α – primitivní prvek Galoisova tělesa GF(16). Galoisovo těleso je tvořeno rozšířením tělesa polynomů pomocí nerozložitelného (ireducibilního) polynomu.

nerozložitelný polynom

Definice: Binární cyklický kód K délky n má generující kořeny $\alpha^{i_1}, \alpha^{i_2}, \ldots, \alpha^{i_s}$, jestliže α je prvek řádu n v některém rozšíření tělesa polynomů. Pro každý polynom $v(x)$, který je kódovým slovem kódování K platí $v(\alpha^{i_1}) = 0$. Polynom $v(x)$ je úplně popsán pomocí kořenů generujícího polynomu.

Například: Hammingovy kódy jsou binární cyklické kódy délky $2^m - 1$ s jediným generujícím kořenem α. Jestliže zvolíme primitivní prvek α v Galoisově tělese GF(2^m), pak minimální polynom prvku je polynom q(x) stupně m. Tento polynom q(x) je generujícím polynomem kódu s m kontrolními znaky. Hammingovým kódem je pak ($2^m - 1$, $2^m - m - 1$)-kód.

BCH kódy jsou pak těmito nástroji definovány jako zobecnění Hammingových kódů. Zatímco pro opravu jednoduché chyby má kontrolní matice **H** jeden řádek [nad Galoisovým tělesem GF(16)], pro dvojnásobné chyby pak má dva řádky. Pro kód o délce 15 je to matice:

$$\mathbf{H} = \begin{bmatrix} 1 & \alpha & \alpha^2 & \ldots & \alpha^{14} \\ 1 & \alpha^3 & (\alpha^3)^2 & \ldots & (\alpha^3)^{14} \end{bmatrix}. \tag{100}$$

Tuto matici můžeme přepsat pomocí tabulky převodů do binární matice s osmi řádky. Kódová slova vyhovují rovnici:

$$\begin{bmatrix} 1 & \alpha & \alpha^2 & \ldots & \alpha^{14} \\ 1 & \alpha^3 & (\alpha^3)^2 & \ldots & (\alpha^3)^{14} \end{bmatrix} \cdot \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \ldots \\ w_{13} \\ w_{14} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \qquad (101)$$

kontrolní matice

Kontrolní matice složená z prvků α^i Galoisova tělesa GF(16) je převoditelná do binárních polynomů:

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \qquad (102)$$

Tento BCH-kód je (15, 7)-kód, neboť matice **H** má lineárně nezávislé řádky. Jestliže α má minimální polynom $\mathbf{M}_1(x) = x^4 + x + 1$ a α^3 má minimální polynom $\mathbf{M}_3(x) = x^4 + x^3 + x^2 + 1$. Potom každé kódové slovo $\mathbf{w}(x)$ je představováno polynomem, který je dělitelný jak polynomem $\mathbf{M}_1(x)$, tak polynomem $\mathbf{M}_3(x)$. Poslední polynom je nerozložitelný, to znamená, že kódové slovo musí být dělitelné součinem polynomů $\mathbf{M}_1(x)\mathbf{M}_3(x)$. Tento kód opravuje dvojnásobné chyby.

generující polynom

Polynom s nejnižším stupněm, který představuje nenulové slovo, je ***generujícím polynomem*** našeho kódu:

$$\begin{aligned} \mathbf{g}(x) &= \mathbf{M}_1(x)\mathbf{M}_3(x) \\ &= (x^4 + x + 1)(x^4 + x^3 + x^2 + 1) \\ &= x^8 + x^7 + x^6 + x^4 + 1 \end{aligned} \qquad (103)$$

Všechna kódová slova BCH-kódu jsou násobky polynomů $q(x)g(x)$. Pro (15, 7)-kód je generující polynom $g(x) = x^8 + x^7 + x^6 + x^4 + 1$ násoben polynomem $q(x)$, který musí být stupně nižšího než 7.

Generující polynom $g(x) = \mathrm{LCM}[M_1(x), M_2(x), \ldots, M_{2t}(x)]$ BCH-kódu je nejmenším společným násobkem (LCM = Least Common Multiple) všech minimálních polynomů $M_i(x)$. Informační znaky potom kódujeme násobením polynomem $g(x)$ nebo (při čtení pozpátku) dělením tímto polynomem.

Příklad 7.1: použijeme-li BCH-kód délky 15 s plánovanou vzdáleností 7, generující polynom bude například:

$$g(x) = (x^4 + x + 1)(x^4 + x^3 + x^2 + 1)(x^2 + x + 1) = \\ = x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1 \qquad (104)$$

Informační znaky 01001 zakódujeme následovně:

$$g(x)(x + x^4) = (x^{14} + x^{12} + x^{11} + x^8 + x^4 + x^3 + x^2 + x)$$

a získáme kódové slovo 011110001001101.

Pro dekódování BCH-kódů sestavíme kontrolní matici. Z generující matice (100) můžeme vypočítat:

$$\mathbf{H}\begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \dots \\ w_{n-2} \\ w_{n-1} \end{bmatrix} = \begin{bmatrix} w_0 + w_1\alpha + w_2\alpha^2 + \dots + w_{n-1}\alpha^{n-1} \\ w_0 + w_1\alpha^3 + w_2(\alpha^3)^2 + \dots + w_{n-1}(\alpha^3)^{n-1} \end{bmatrix} = \begin{bmatrix} w(\alpha) \\ w(\alpha^3) \end{bmatrix} = \begin{bmatrix} s_1 \\ s_3 \end{bmatrix}. \qquad (105)$$

Přijaté slovo je:

$$\mathbf{w} = [w_0 \quad w_1 \quad \dots \quad w_{14}], \qquad (106)$$

které se od vyslaného slova liší na *i*-tém a *j*-tém místě. Chybové slovo (rozdíl vyslaného a přijatého slova) pak je:

$$\mathbf{e} = 00\dots010\dots010\dots00, \qquad (107)$$

nebo ve tvaru polynomu:

$$\mathbf{e} = z^i + z^j. \qquad (108)$$

Známe jeho syndrom:

$$\begin{bmatrix} s_1 \\ s_3 \end{bmatrix} = \mathbf{H}\mathbf{w}^T = \mathbf{H}\mathbf{e}^T = \begin{bmatrix} \alpha^i + \alpha^j \\ \alpha^{3i} + \alpha^{3j} \end{bmatrix}. \qquad (109)$$

Chceme tedy určit čísla *i* a *j*, pro která platí:

$$\begin{aligned} s_1 &= \alpha^i + \alpha^j \\ s_3 &= \alpha^{3i} + \alpha^{3j} \end{aligned}. \qquad (110)$$

Z první rovnice dostáváme:

$$\begin{aligned} s_1^3 &= \left(\alpha^i + \alpha^j\right)^3 = \alpha^{3i} + \alpha^{2i}\alpha^j + \alpha^i\alpha^{2j} + \alpha^{3j} = \\ &= s_3 + \alpha^{2i}\alpha^j + \alpha^i\alpha^{2j} = s_3 + \alpha^i\alpha^j\left(\alpha^i + \alpha^j\right) = \\ &= s_3 + s_1\alpha^i\alpha^j . \end{aligned} \tag{111}$$

Známe tedy součet

$$\alpha^i + \alpha^j = s_1 . \tag{112}$$

i součin

$$\alpha^i\alpha^j = \frac{s_1^3 - s_3}{s_1} = s_1^2 + s_3 s_1^{-1} \tag{113}$$

neznámých veličin α^i a α^j. Tyto veličiny jsou kořeny kvadratické rovnice:

$$\lambda^2 + s_1\lambda + \left(s_1^2 + s_3 s_1^{-1}\right) = 0 . \tag{114}$$

Pro řešení této rovnice dosazujeme postupně $\lambda = 1, \alpha, \alpha^2, \ldots$ (obvyklý vzorec pro řešení kvadratických rovnic samozřejmě nad konečným tělesem neplatí). Tak získáme kořeny α^i a α^j a opravíme následně prvky přijatého slova w_i a w_j.

Poznámka: Pokud při přenosu došlo pouze k jednoduché chybě, pak jeden z kořenů rovnice (114) bude nulový. Pokud nedošlo k žádné chybě, bude syndrom přijatého slova nulový vektor a řešení rovnice (114) není potřeba, resp. má dvojnásobný nulový kořen.

Oprava dvou chyb u BCH kódů vede na formulaci a řešení kvadratické rovnice. Jejich minimální vzdálenost musí být nejméně $d = 5$. Pokud srovnáme BCH-kód s rozšířeným Hammingovým kódem, schopným opravit jednoduchou chybu a detekovat až trojnásobnou chybu, mají BCH-kódy příznivější vlastnosti (menší redundanci).

7.1 BCH-kód s plánovanou vzdáleností

Binární BCH-kód s plánovanou vzdáleností $d = 2, 3, 4, \ldots$ je kód liché délky $n \geq d$ s generujícími kořeny $\alpha, \alpha^2, \alpha^3, ..., \alpha^{d-1}$, kde α je prvek n-tého řádu v tělese GF(2^m).

Poznámka: Sudé mocniny mezi generujícími kořeny jsou nadbytečné, protože polynom $v(z) \in Z_2^{(n)}$ má s každým kořenem α^i také kořeny α^{2i}, α^{4i}, … Odtud plyne, že BCH-kódy s plánovanou vzdáleností $d = 2t$ nebo $d = 2t + 1$ jsou totožné a mají generující kořeny $\alpha, \alpha^3, \alpha^5, ..., \alpha^{2t-1}$, proto budeme dále vycházet z předpokladu, že vzdálenost d je lichá.

BCH-kód s plánovanou vzdáleností *d* můžeme také definovat pomocí kontrolní matice:

$$\mathbf{H} = \begin{bmatrix} 1 & \alpha & \alpha^2 & \alpha^3 & \dots & \alpha^{n-1} \\ 1 & \alpha^3 & (\alpha^3)^2 & (\alpha^3)^3 & \dots & (\alpha^3)^{n-1} \\ 1 & \alpha^5 & (\alpha^5)^2 & (\alpha^5)^3 & \dots & (\alpha^5)^{n-1} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & \alpha^{d-2} & (\alpha^{d-2})^2 & (\alpha^{d-2})^3 & \dots & (\alpha^{d-2})^{n-1} \end{bmatrix} \tag{115}$$

Počet kontrolních znaků BCH-kódu je $n-k \le \frac{1}{2}m(d-1)$. Kontrolní matice má $\frac{1}{2}(d-1)$ řádků nad tělesem GF(2^m), a proto má $\frac{1}{2}m(d-1)$ řádků nad tělesem Z_2. Pokud jsou tyto řádky lineárně nezávislé, platí $n-k=\frac{1}{2}m(d-1)$, pokud jsou závislé, bude $n-k$ menší.

Příklad 7.2:
BCH-kód pro $d = 3$. Jde o binární kód s generujícím kořenem α. Pokud $n = 2^m - 1$, je α primitivní prvek tělesa GF(2^m), takže BCH-kód je Hammingův ($2^m - 1$, $2^m - m - 1$)-kód a platí $n - k = m$.

Příklad 7.3:
BCH-kód pro $d = 5$. Jde o BCH-kód opravující dvojnásobné chyby, který byl ukázán v předchozím textu. Pro $n = 2^m - 1$, platí $n - k = m$.

Příklad 7.4:
BCH-kód pro $d = 7$. Určíme BCH-kód délky 15. Ten má generující kořeny α, α^3, α^5, kde α je primitivní prvek tělesa GF(16). Platí:

$$\begin{aligned} &M_1(x) = x^4 + x + 1, \\ &M_3(x) = x^4 + x^3 + x^2 + x + 1, \\ &M_5(x) = x^2 + x + 1. \end{aligned} \tag{116}$$

generující polynom je tedy:

$$\begin{aligned} g(x) &= M_1(x)M_3(x)M_5(x) = \\ &= (x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1)(x^2 + x + 1) = \\ &= x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1. \end{aligned} \tag{117}$$

Jedná se o (15, 5)-kód s 10 kontrolními znaky, takže $n-k \le \frac{1}{2}m(d-1)$ a kód opravuje trojnásobné chyby.

maticová metoda dekódování

Pro dekódování můžeme použít ***maticovou metodu dekódování***. BCH-kód s plánovanou vzdáleností $2t + 1$ je schopen opravit t-násobné chyby. Abychom je opravili, najdeme částečné dekódování $\delta : Z_2^{(n)} \to K$ takové, že kdykoliv při vyslání kódového slova **v** dojde k t-násobné chybě, potom přijaté slovo **w** splňuje $\delta(\mathbf{w}) = \mathbf{v}$. Stačí ovšem najít chybové slovo

$$\mathbf{e} = \mathbf{w} - \mathbf{v} \tag{118}$$

a položit $\delta(\mathbf{w}) = \mathbf{w} - \mathbf{e}$.

Pro každé slovo $\mathbf{w} = w_0 w_1 \ldots w_{n-1}$ pak nastávají tyto možnosti:

1) Došlo k t-násobné chybě, existuje tedy slovo **e** váhy $\leq t$ takové, že **w** – **e** je kódové slovo. pak ze skutečnosti, že minimální vzdálenost kódu je alespoň $2t + 1$ plyne, že takové slovo existuje pouze jedno. Dekódování tedy určí chybové slovo **e**.

2) Došlo k chybě více než t-násobné. Tuto situaci naše dekódování někdy objeví, místo určení chybového slova **e** pak bude ohlášena velká chyba. V některých případech ale taková chyba vůbec nebude objevena a dekódování skončí nalezením nesprávného slova **e**.

lokátor chyb

Předpokládáme, že došlo k t-násobné chybě. Hledané chybové slovo **e** má váhu p ($\leq t$), takže existují souřadnice $e_{i_1}, e_{i_2}, ..., e_{i_p}$ rovné 1, zatímco ostatní souřadnice mají hodnotu 0. Určíme polynom $f(x)$, tzv. ***lokátor chyb***, jehož kořeny jsou $\alpha^{-i_1}, \alpha^{-i_2}, ..., \alpha^{-i_p}$. Prozkoumáním kořenů tohoto polynomu pak určíme, která místa přijatého slova je třeba opravit: opravíme w_i, právě když $f(\alpha^{-i}) = 0$.

Dekódování přijatého slova pak má celkem tři kroky:

1) Určení syndromu přijatého slova.

$$\mathbf{s}^T = \mathbf{H}\mathbf{w}^T = \begin{bmatrix} 1 & \alpha & \alpha^2 & \alpha^3 & \ldots & \alpha^{n-1} \\ 1 & \alpha^2 & (\alpha^3)^2 & (\alpha^3)^3 & \ldots & (\alpha^3)^{n-1} \\ 1 & \alpha^5 & (\alpha^5)^2 & (\alpha^5)^3 & \ldots & (\alpha^5)^{n-1} \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 1 & \alpha^{2t} & (\alpha^{2t})^2 & (\alpha^{2t})^3 & \ldots & (\alpha^{2t})^{n-1} \end{bmatrix} \cdot \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \ldots \\ w_{n-1} \end{bmatrix} = \tag{119}$$

$$= \begin{bmatrix} w_0 + w_1\alpha + w_2\alpha^2 + \ldots + w_{n-1}\alpha^{n-1} \\ w_0 + w_1\alpha^2 + w_1(\alpha^2)^2 + \ldots + w_{n-1}(\alpha^2)^{n-1} \\ w_0 + w_1\alpha^3 + w_1(\alpha^3)^2 + \ldots + w_{n-1}(\alpha^3)^{n-1} \\ \ldots \\ w_0 + w_1\alpha^{2t} + w_1(\alpha^{2t})^2 + \ldots + w_{n-1}(\alpha^{2t})^{n-1} \end{bmatrix} = \begin{bmatrix} w(\alpha) \\ w(\alpha^2) \\ w(\alpha^3) \\ \ldots \\ w(\alpha^{2t}) \end{bmatrix}.$$

2) Určení koeficientů lokátoru chyb $f(x)$. Tím je polynom:

$$f(x) = (1 - a_1 x)(1 - a_2 x)...(1 - a_p x)\,, \tag{120}$$

jakmile určíme tento polynom a jeho kořeny, což jsou prvky $a_n^{-i} = \alpha^{-i_n}$ pro $n = 1, 2, \ldots, p$, máme určeno chybové slovo **e**:

$$e_i = \begin{cases} 1 & \text{pokud } f(\alpha^{-i}) = 0, \\ 0 & \text{jinak.} \end{cases} \tag{121}$$

polynom $f(x) = f_0 + f_1 x + ... + f_p x^p$ má první koeficient $f_0 = 1$ (protože $f(0) = 1$) a ostatní koeficienty určíme ze soustavy lineárních rovnic. K jejímu odvození použijeme mocninnou řadu:

$$S(x) = \sum_{k=1}^{\infty} s_k x^k\,, \tag{122}$$

kde koeficienty jsou:

$$s_k = a_1^k + a_2^k + ... + a_p^k = \sum_{n=1}^{p} a_n^k\,. \tag{123}$$

Známe tedy pouze koeficienty $s_1, s_2, \ldots, s_{2t}$, ale to nám postačuje. Pro koeficienty lokátoru chyb platí rovnice:

$$\begin{array}{lllllll} f_1 & & & & & & = s_1 \\ s_2 f_1 & + s_1 f_2 & + f_3 & & & & = s_3 \\ s_4 f_1 & + s_3 f_2 & + s_2 f_3 & + s_1 f_4 & + f_5 & & = s_5 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \\ s_{2p-2} f_1 & + s_{2p-3} f_2 & + s_{2p-4} f_3 & + \ldots & + s_p f_{p-1} + s_{p-1} f_p & & = s_{2p-1} \end{array} \tag{124}$$

3) Oprava chyb nalezením kořenů polynomu $f(x)$ postupným dosazováním prvků 1, $\alpha^{-1}, \alpha^{-2}, ...$, a opravíme i-tý znak slova **w**, když platí $f\left(\alpha^{-i}\right) = 0$.

7.2 Použití BCH-kódů

V praxi se často používají BCH-kódy, které opravují dvojnásobné chyby a současně detekují trojnásobné chyby. Jejich minimální vzdálenost musí být alespoň 6. Nejčastěji má kontrolní matice tvar:

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & \alpha & \alpha^2 & \alpha^3 & \dots & \alpha^{n-1} \\ 1 & \alpha^3 & \alpha^6 & \alpha^9 & \dots & \alpha^{3(n-1)} \end{bmatrix}, \tag{125}$$

kde je α primitivní prvek Galoisova tělesa GF(2^m) a výrazy α^i jsou mocniny tohoto primitivního prvku pro $i = 0, 1, 2, \dots, 2^m - 2$.

Jednotlivé řádky kontrolní matice **H** mají následující význam: součin s přijatým slovem **w** je testem, zda 1, α, α^3 jsou kořeny polynomu odpovídajícího tomuto vektoru. První řádek kontrolní matice tedy říká, že α^1 a tedy také α^2, α^4, α^8, … jsou kořeny kódových polynomů. Poslední řádek zaručuje, že α^3, α^6, α^{12}, … jsou také kořeny kódových polynomů. A protože α^0, α^1, α^2, α^3, α^4 jsou pěticí po sobě jdoucích mocnin α, které jsou kořeny kódových polynomů, je $d_{\min} \geq 6$. Kontrolní matice **H** bude po dosazení za prvky α^i obsahovat jen nuly a jedničky.

Například kontrolní matice BCH-kódu (15, 6)-kódu, která má jako polynomy $g_0(x) = x + 1$, $g_1(x) = x^4 + x + 1$, $g_3(x) = x^4 + x^3 + x^2 + x + 1$. Generující polynom BCH-kódu je nejmenším společným násobkem těchto minimálních polynomů:

$$\begin{aligned} g(x) &= g_0(x)g_1(x)g_3(x) = \\ &= (x + 1)(x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1) = \\ &= x^9 + x^6 + x^5 + x^4 + x + 1. \end{aligned} \tag{126}$$

Kontrolní matice pak má, po vynechání prvních tří nulových řádků, tvar:

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{127}$$

Přijaté slovo může mít chyby v *i*-tém a *j*-tém bitu. Vynásobením přijatého slova kontrolní maticí **H** získáme syndrom **s** = [s_0, s_1, s_3], pro jehož složky platí:

$$s_0 = 0$$
$$s_1 = \alpha^i + \alpha^j, \tag{128}$$
$$s_3 = \alpha^{3i} + \alpha^{3j}.$$

Umocněním druhé rovnice na třetí a eliminací α^j postupně rovnici zjednodušíme:

$$s_1^3 = \alpha^{3i} + \alpha^{2i}\alpha^j + \alpha^i\alpha^{2j} + \alpha^{3j} = s_3 + \alpha^i\alpha^j s_1 = s_3 + \alpha^i s_1\left(\alpha^i + s_1\right) \tag{129}$$

Nyní již můžeme sestavit rovnici:

$$\alpha^{2i}s_1 + \alpha^i s_1^2 + s_1^3 + s_3 = 0. \tag{130}$$

Jejím řešením určíme polohu prvního chybného *i*-tého bitu. Polohu druhého chybného *j*-tého bitu zjistíme z rovnice $2\alpha^i + s_1 = \alpha^j$. Prvky α^i a α^j jsou lokátory chyb, protože svojí polohou v kontrolní matici ukazují polohu chybných bitů. Pokud je $s_0 \neq 0$ a $\alpha^i + s_1 = 0$, došlo pouze k jednoduché chybě na *i*-té pozici.

7.3 Galoisova tělesa

těleso

Těleso (anglicky division ring) je algebraická struktura, na které jsou definovány dvě binární operace (sčítání – $\oplus$ a násobení – $\otimes$). Je rozšířením okruhu, oproti kterému navíc přináší existenci inverzního prvku pro obě binární operace (okruh vyžadoval existenci inverzního prvku jen pro operaci $\oplus$). Běžně pracujeme s tělesem **R**, které je tvořeno reálnými čísly a definicí jejich sčítání a násobení, tělesem **C**, tvořeným komplexními čísly a jejich sčítáním a násobením nebo tělesem **Q**, které tvoří racionální čísla a definice jejich sčítání a násobení. Všechna tato tělesa mají nekonečný počet prvků, proto je nutné existenci těles ověřovat.

konečné těleso

Galoisovo těleso

Pro popisy kódů jsou významná především ***konečná tělesa***, nazývaná ***Galoisova tělesa***, není nutné axiómy ověřovat, stačí znát popis všech těles. Galoisovo těleso je určeno počtem svých prvků q a označuje se GF(q). Všechna Galoisova tělesa musí mít definován nulový a jednotkový prvek. Operace s nimi mohou být definovány např. tabulkou. Pro GF(2) to budou tabulky:

$\oplus$	0	1
0	0	1
1	1	0

$\otimes$	0	1
0	0	0
1	0	1

Pro naše potřeby jsou důležitá zejména Galoisova tělesa s algebraickým rozšířením. Pro každé těleso **T** a každý polynom q(x) stupně n je definován okruh **T**/q(x), který nazýváme ***algebraické rozšíření tělesa*** **T**. Prvky tohoto okruhu jsou všechny polynomy nad tělesem **T** stupně většího nebo rovného n –

1. Algebraické rozšíření **T**/q(x) splňuje podmínky toho, aby bylo tělesem právě tehdy, když je polynom q(x) nerozložitelný (ireducibilní).

Příklad 7.5:

Uvažujeme těleso Z_2, které bude rozšířeno polynomem $q(x) = x^2 + x + 1$. Tato nová nula, kterou označíme α, je definována rovnicí $\alpha^2 + \alpha + 1 = 0$, platí tedy $\alpha^2 = \alpha + 1$. Těleso vzniklé tímto rozšířením tedy bude mít dva nové prvky α a $\alpha + 1$. Celkově tedy máme čtyři prvky, pro které definujeme operace sčítání a násobení:

$\oplus$	0	1	α	$\alpha + 1$
0	0	1	α	$\alpha + 1$
1	$\alpha + 1$	0	1	α
α	α	$\alpha + 1$	0	1
$\alpha + 1$	1	α	$\alpha + 1$	0

$\otimes$	0	1	α	$\alpha + 1$
0	0	0	0	0
1	0	1	α	$\alpha + 1$
α	0	α	$\alpha + 1$	1
$\alpha + 1$	0	$\alpha + 1$	1	α

Dále jsou uvedeny tabulky Galoisových těles

$$GF(p^n) = Z_p/q(x),$$

kde je p – prvočíslo, $n > 1$, q(x) – ireducibilní polynom stupně n nad Z_p. Každý prvek vyjádříme jako polynom α proměnné stupně $< n$ a zároveň (kromě nuly) jako mocnina prvku α. Tím je umožněno snadné sčítání (obyčejné sčítání polynomů) i násobení ($\alpha^i\alpha^j = \alpha^{i+j}$). Polynomy q($x$) volíme tak, že prvek α je primitivním prvkem tělesa $GF(p^n)$.

$GF(4) = Z_2/(x^2 + x + 1)$	$GF(8) = Z_2/(x^3 + x + 1)$
0	0
$1 = \alpha^0 = \alpha^3$	$1 = \alpha^0 = \alpha^7$
$\alpha = \alpha^1$	$\alpha = \alpha^1$
$1 + \alpha = \alpha^2$	$\alpha^2 = \alpha^2$
	$1 + \alpha = \alpha^3$
	$\alpha + \alpha^2 = \alpha^4$
	$1 + \alpha + \alpha^2 = \alpha^5$
	$1 + \alpha^2 = \alpha^6$

$GF(9) = Z_3/(x^2 + x + 2)$	$GF(16) = Z_2/(x^4 + x + 1)$
0	0
$1 = \alpha^0 = \alpha^8$	$1 = \alpha^0 = \alpha^{15}$
$\alpha = \alpha^1$	$\alpha = \alpha^1$
$1 + 2\alpha = \alpha^2$	$\alpha^2 = \alpha^2$
$2 + 2\alpha = \alpha^3$	$\alpha^3 = \alpha^3$
$2 = \alpha^4$	$1 + \alpha = \alpha^4$
$2\alpha = \alpha^5$	$\alpha + \alpha^2 = \alpha^5$
$2 + \alpha = \alpha^6$	$\alpha^2 + \alpha^3 = \alpha^6$
$1 + \alpha = \alpha^7$	$1 + \alpha + \alpha^3 = \alpha^7$
	$1 + \alpha^2 = \alpha^8$
	$\alpha + \alpha^3 = \alpha^9$
	$1 + \alpha + \alpha^2 = \alpha^{10}$
	$\alpha + \alpha^2 + \alpha^3 = \alpha^{11}$
	$1 + \alpha + \alpha^2 + \alpha^3 = \alpha^{12}$
	$1 + \alpha^2 + \alpha^3 = \alpha^{13}$
	$1 + \alpha^3 = \alpha^{14}$

$GF(25) = Z_5/(x^2 + x + 2)$	$GF(27) = Z_3/(x^3 + 2x + 1)$
0	0
$1 = \alpha^0 = \alpha^{24}$	$1 = \alpha^0 = \alpha^{26}$
$\alpha = \alpha^1$	$\alpha = \alpha^1$
$3 + 4\alpha = \alpha^2$	$\alpha^2 = \alpha^2$
$2 + 4\alpha = \alpha^3$	$2 + \alpha = \alpha^3$
$2 + 3\alpha = \alpha^4$	$2\,\alpha + \alpha^2 = \alpha^4$
$4 + 4\alpha = \alpha^5$	$2 + \alpha + 2\alpha^2 = \alpha^5$
$2 = \alpha^6$	$1 + \alpha + \alpha^2 = \alpha^6$
$2\alpha = \alpha^7$	$2 + 2\alpha + \alpha^2 = \alpha^7$
$1 + 3\alpha = \alpha^8$	$2 + 2\alpha^2 = \alpha^8$
$4 + 3\alpha = \alpha^9$	$1 + \alpha = \alpha^9$
$4 + \alpha = \alpha^{10}$	$\alpha + \alpha^2 = \alpha^{10}$
$3 + 3\alpha = \alpha^{11}$	$2 + \alpha + \alpha^2 = \alpha^{11}$
$4 = \alpha^{12}$	$\alpha + \alpha^2 = \alpha^{12}$
$4\alpha = \alpha^{13}$	$2 = \alpha^{13}$
$2 + \alpha = \alpha^{14}$	$2\alpha = \alpha^{14}$
$3 + \alpha = \alpha^{15}$	$2\alpha^2 = \alpha^{15}$
$3 + 2\alpha = \alpha^{16}$	$1 + 2\alpha = \alpha^{16}$
$1 + \alpha = \alpha^{17}$	$\alpha + 2\alpha^2 = \alpha^{17}$
$3 = \alpha^{18}$	$1 + 2\alpha + \alpha^2 = \alpha^{18}$
$3\alpha = \alpha^{19}$	$2 + 2\alpha + 2\alpha^2 = \alpha^{19}$
$4 + 2\alpha = \alpha^{20}$	$1 + \alpha + 2\alpha^2 = \alpha^{20}$
$1 + 2\alpha = \alpha^{21}$	$1 + \alpha^2 = \alpha^{21}$
$1 + 4\alpha = \alpha^{22}$	$2 + 2\alpha = \alpha^{22}$
$2 + 2\alpha = \alpha^{23}$	$2\alpha + \alpha^2 = \alpha^{23}$
	$1 + 2\alpha + 2\alpha^2 = \alpha^{24}$
	$1 + 2\alpha^2 = \alpha^{25}$

$GF(32) = Z_2/(x^5 + x^2 + 1)$

0	0
$1 = \alpha^0 = \alpha^{31}$	$1 + \alpha + \alpha^2 + \alpha^3 + \alpha^4 = \alpha^{15}$
$\alpha = \alpha^1$	$1 + \alpha + \alpha^3 + \alpha^4 = \alpha^{16}$
$\alpha^2 = \alpha^2$	$1 + \alpha + \alpha^4 = \alpha^{17}$
$\alpha^3 = \alpha^3$	$1 + \alpha = \alpha^{18}$
$\alpha^4 = \alpha^4$	$\alpha + \alpha^2 = \alpha^{19}$
$1 + \alpha^2 = \alpha^5$	$\alpha^2 + \alpha^3 = \alpha^{20}$
$\alpha + \alpha^3 = \alpha^6$	$\alpha^3 + \alpha^4 = \alpha^{21}$
$\alpha^2 + \alpha^4 = \alpha^7$	$1 + \alpha^2 + \alpha^4 = \alpha^{22}$
$1 + \alpha^2 + \alpha^3 = \alpha^8$	$1 + \alpha + \alpha^2 + \alpha^3 = \alpha^{23}$
$\alpha + \alpha^3 + \alpha^4 = \alpha^9$	$\alpha + \alpha^2 + \alpha^3 + \alpha^4 = \alpha^{24}$
$1 + \alpha^4 = \alpha^{10}$	$1 + \alpha^3 + \alpha^4 = \alpha^{25}$
$1 + \alpha + \alpha^1 = \alpha^{11}$	$1 + \alpha + \alpha^2 + \alpha^4 = \alpha^{26}$
$\alpha + \alpha^2 + \alpha^3 = \alpha^{12}$	$1 + \alpha + \alpha^3 = \alpha^{27}$
$\alpha^2 + \alpha^3 + \alpha^4 = \alpha^{13}$	$\alpha + \alpha^2 + \alpha^4 = \alpha^{28}$
$1 + \alpha^2 + \alpha^3 + \alpha^4 = \alpha^{14}$	$1 + \alpha^3 = \alpha^{29}$
	$\alpha + \alpha^4 = \alpha^{30}$

Kontrolní otázky:

25. Co je konečné těleso?
26. Uveďte tabulku Galoisova tělesa $GF(8) = Z_2/(x^3 + x + 1)$?
27. Jak jsou definovány BCH-kódy?
28. Jak probíhá kódování BCH-kódu?
29. Jak probíhá dekódování BCH-kódu?

Korespondenční úkol:

3. Zvolte si vhodný BCH-kód a ukažte jak s jeho pomocí zakódovat informační slovo, jak jej dekódovat při bezchybném přenosu i v případě jeho poškození jednoduchou i vícenásobnou chybou.

Shrnutí obsahu kapitoly

V této kapitole jsme se seznámili s významnou třídou BCH-kódů. Seznámili jsme se s algebraickými strukturami potřebnými pro jejich definici, zejména s Galoisovými tělesy. Umíme sestavit BCH-kód i popsat algoritmus jeho dekódování včetně opravy chyb.

8 Reedovy-Solomonovy kódy

Cíl:

Cílem celé kapitoly je rozšířit znalosti BCH-kódů o BCH-kódy nad q-znakovou abecedou GF(q), kde je q mocnina prvočísla. Jejich vlastnosti jsou analogické s binárnímu BCH-kódy, ale umožňují nám lépe definovat vlastnosti kódů, které nazýváme Reedovy-Solomonovy kódy.

Po jejím prostudování byste měli být schopni:

- Sestavit Reedův-Solomonův kód.
- Vytvářet dobré binární kódy.
- Dekódovat slovo zabezpečené Reedovým-Solomonovým kódem.
- Popsat jak jsou zabezpečeny optické paměti.

Klíčová slova této kapitoly:

BCH-kód v užším smyslu, kontrolní matice, primitivní prvek, Reedovy-Solomonovy kódy, shluková chyba, doprovodná matice, optická paměť, kód SBC-DBD, optická paměť, jamka, země, RLL-kód, symbol, rámec, EFM-kód.

Doba potřebná ke studiu: 4 hodiny

Průvodce studiem

Studium této kapitoly je nutné k pochopení konstrukce dobrých binárních kódů, které jsou rozšířením BCH-kódů. Reedovy-Solomonovy kódy umožňují sestavit kódy schopné indikovat odstranit shlukové chyby a jsou tak výhodné k použití při zabezpečení např. optických pamětí.
Na studium této části si vyhraďte čtyři hodiny. Zkuste si samostatně vyřešit příklady tvorby RS-kódů a jejich použití.

Zatím jsme se zabývali jen binárními BCH-kódy. Tato kapitola pracuje s BCH-kódy nad q-znakovou abecedou GF(q), kde je q mocnina prvočísla. I pro $q = 2$ je jejich definice obecnější, než jak byla uvedena v předchozí kapitole. Jejich vlastnosti jsou analogické s binárnímu BCH-kódy. Volíme vzdálenost d a jejich konstrukce zajistí že $d_{\min} \geq$ d. Takže můžeme opravit t-násobné chyby pro $d = 2t + 1$.

Definice: q-znakový BCH-kód s plánovanou vzdáleností $d = 2, 3, 4, \ldots$ je kód délky $n \geq d$, kde je n nesoudělné s q, který má generující kořeny:

$$\alpha^{j}, \alpha^{j+1}, \alpha^{j+2}, ..., \alpha^{j+d-2} \text{ (pro některé } j = 0, 1, \ldots, n - d + 1), \qquad (131)$$

kde je α – prvek n-tého řádu v tělese GF(q^m).

Pokud je $j = 1$, jsou generující kořeny $\alpha, \alpha^2, \alpha^3, ..., \alpha^{d-1}$, mluvíme o ***BCH-kódu v užším smyslu***. Minulá kapitola se tedy zabývala BCH-kódy v užším smyslu.

BCH-kód užším smyslu

Z nesoudělnosti čísel n a q vyplývá, že prvek α vždy existuje. ***Kontrolní matice*** BCH-kódu je:

kontrolní matice

$$\mathbf{H} = \begin{bmatrix} 1 & \alpha^{j} & (\alpha^{j})^2 & (\alpha^{j})^3 & ... & (\alpha^{j})^{n-1} \\ 1 & \alpha^{j+1} & (\alpha^{j+1})^2 & (\alpha^{j+1})^3 & ... & (\alpha^{j+1})^{n-1} \\ 1 & \alpha^{j+2} & (\alpha^{j+2})^2 & (\alpha^{j+2})^3 & ... & (\alpha^{j+2})^{n-1} \\ ... & ... & ... & ... & ... & ... \\ 1 & \alpha^{j+d-2} & (\alpha^{j+d-2})^2 & (\alpha^{j+d-2})^3 & ... & (\alpha^{j+d-2})^{n-1} \end{bmatrix}. \qquad (132)$$

Příklad 8.1: Máme binární BCH-kód délky 7 s generujícími kořeny 1 a α. Volíme primitivní prvek tělesa GF(8). Potom α má minimální polynom $M_1(x) = x^3 + x + 1$ a 1 má minimální polynom $M_0(x) = x - 1 = x + 1$. Generující polynom kódu je:

$$g(x) = (x+1)(x^3 + x + 1), \qquad (133)$$

primitivní prvek

Jedná se o zmenšený Hammingův kód, kódová slova jsou v(z), pro která platí v(1) = 0 (nebo-li $v_0v_1\ldots v_{n-1}$ má sudou paritu) a v(α) = 0 (čili $v_0v_1\ldots v_{n-1}$ patří k Hammingovu kódu).

Hammingův kód

Příklad 8.2: Ternární BCH-kód délky 8 s generujícími kořeny α^2, α^3, α^4, α^5. Volíme α primitivní prvek tělesa GF(9). Potom prvek α a také α^3 má minimální polynom $x^2 + x + 2$. Prvek α^2 má minimální polynom:

$$M_2(x) = (x - \alpha^2)(x - \alpha^6) = x^2 + 1. \qquad (134)$$

Prvek $\alpha^4 = 2$ má minimální polynom $M_4(x) = x - 2 = x + 2$ a prvek α^5 má minimální polynom:

$$M_5(x) = (x - \alpha^5)(x - \alpha^7) = x^2 + 2x + 2. \qquad (135)$$

Generující polynom kódu je pak:

$$g(x) = (x^2 + 1)(x^2 + x + 2)(x + 2)(x^2 + 2x + 2). \quad (136)$$

Zjišťujeme, že pouze jeden znak je informační, takže jde o opakovací kód.

Příklad 8.3: Ternární BCH-kód délky 8 s generujícími kořeny 1, α, α^2, α^3. Volíme opět α primitivní prvek v GF(9). Generující polynom je v tomto případě:

$$g(x) = M_0(x)M_1(x)M_2(x) = (x-1)(x^2 + x + 2)(x^2 + 1). \quad (137)$$

Takže se jedná o (8-3) kód opravující dvojnásobné chyby.

Příklad 8.4: čtyřznakový BCH-kód délky 3 s generujícím kořenem α^2. Kódovou abecedu volíme ve tvaru: GF(4) = {0, 1, *z*, *t*}, kde je $t = z^2 = 1 + z$. Protože prvek *z* má řád 3, můžeme volit $z = \alpha$. Minimální polynom prvku $\alpha^2 = t$ je $x - t$ (= $x + t$), proto:

$$g(x) = (x + t). \quad (138)$$

Jedná se o (3, 2)-kód s generující maticí:

$$\mathbf{G} = \begin{bmatrix} t & 1 & 0 \\ 0 & t & 1 \end{bmatrix}, \quad (139)$$

a jedná se o Reedův-Solomonův kód.

Reedovy-Solomonovy kódy

Reedovy-Solomonovy kódy (RS-kódy) jsou speciální BCH-kódy délky $q - 1$. Jsou tak důležité, protože mají největší myslitelnou minimální vzdálenost a hlavně proto, že s jejich pomocí je možno sestavit dobré binární kódy. Při přenosu jsou znaky takového zdroje vyjádřeny binárními ekvivalenty. Při výskytu shlukové chyby, která způsobí změnu několika po sobě jdoucích bitů, je možné opravit znak vyšší abecedy. V příznivém případě je možno opravit celou shlukovou chybu (burst-error). Název kódu je odvozen od iniciál jejich tvůrců, kterými jsou Irving S. Reed a Gustave Solomon, kteří je publikovali v roce 1960.

Například kód z posledního příkladu je RS-kód (3, 2)-kód. Sedmiznakový RS-kód plánované velikosti 3 s generujícími kořeny α, α^2, α^3 můžeme určit tak, že zvolíme:

$$\alpha = 3 \in Z_7, \quad (140)$$

primitivní prvek tělesa Z_7. Generující polynom je pak:

$$g(x) = (x-3)(x-2)(x-6) = x^3 + 2x^2 + x + 6, \quad (141)$$

takže se jedná o (6, 3)-kód s generující maticí:

$$\mathbf{G} = \begin{bmatrix} 6 & 1 & 2 & 1 & 0 & 0 \\ 0 & 6 & 1 & 2 & 1 & 0 \\ 0 & 0 & 6 & 1 & 2 & 1 \end{bmatrix}. \quad (142)$$

Generující polynom g(x) RS-kódu s generujícími kořeny α, α^2, …, α^{2t} je dán výrazem:

$$g(x) = (x-\alpha)(x-\alpha^2)...(x-\alpha^{2t}), \tag{143}$$

a jeho minimální vzdálenost je $d = n - k + 1$. Je to největší možná vzdálenost lineárního (n, k)-kódu.

Příklad 8.5: osmiznakový RS-kód s minimální vzdáleností 5. Označíme $\alpha = z$ primitivní prvek tělesa:

$$GF(8) = Z_2/(x^3 + x + 1). \tag{144}$$

Kód nad abecedou GF(8) s generujícími kořeny α, α^2, α^3, α^4, má generující polynom:

$$g(x) = (x+z)(x+z^2)(x+z^3)(x+z^4) = x^4 + z^3x^3 + z^5x^2 + z, \tag{145}$$

Generující matice je tedy

$$\mathbf{G} = \begin{bmatrix} z & 0 & z^5 & z^3 & 1 & 0 & 0 \\ 0 & z & 0 & z^5 & z^3 & 1 & 0 \\ 0 & 0 & z & 0 & z^5 & z^3 & 1 \end{bmatrix}. \tag{146}$$

8.1 Vytváření dobrých binárních kódů.

V RS-kódu (n, k)-kódu kde je $n = 2^m - 1$. můžeme každý znak nahradit binárním slovem délky $m + 1$. Protože kódová abeceda má $n + 1 = 2^m$ znaků, takže tvoří těleso:

$$GF(2^m) = Z_2 / f(x). \tag{147}$$

Každý znak a je polynomem $a_0 + a_1z + \ldots + a_{m-1}z^{m-1}$ nad tělesem Z_2, který nahradíme slovem délky $m + 1$ tak, že přidáme kontrolu parity:

$$a \leftrightarrow a_0a_1a_2...a_{m-1}a_m, \tag{148}$$

kde je $a_m = a_0 + a_1 + ... + a_{m-1}$ nad Z_2.

Vznikne tak binární (n^*, k^*)-kód, kde je $n^* = n(m+1) = (2^m - 1)(m+1)$ a $k^* = m.k$.

Generující matici sestavíme tak, že každý její řádek v nahradíme řádky:

$$\mathbf{v}, z.\mathbf{v}, z^2.\mathbf{v}, ..., z^{m-1}.\mathbf{v} \tag{149}$$

převedenými do binární podoby. Například abecedu $GF(4) = Z_2/(x^2 + x + 1)$ převádíme:

$$\begin{array}{ll} 0 \leftrightarrow 000 & z \leftrightarrow 011 \\ 1 \leftrightarrow 101 & t \leftrightarrow 110 \end{array}, \tag{150}$$

čtyřznakový (3, 2)-kód, viz (139) převedeme na binární (9, 4)-kód s generující maticí:

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}. \tag{151}$$

Takto vytvořené kódy mají velmi dobré parametry. Minimální vzdálenost d binárního kódu je nejméně dvojnásobkem minimální vzdálenosti RS-kódu:

$$d^* \geq 2(2^m - k + 1). \tag{152}$$

Například z RS-kódu délky 15 a minimální vzdálenosti 6 vznikne binární (75, 40)-kód s minimální vzdáleností 12. Tento kód opravuje pětinásobné chyby (pro srovnání binární BCH-kód délky 63, opravující pětinásobné chyby má 36 informačních bitů).

Poznámka: takto vytvořené binární kódy jsou lineární, ale nemusí být cyklické. Zato jsou schopny opravovat shlukové chyby.

shluková chyba

Shlukovou chybou (shlukem chyb) délky b nazýváme slovo $\mathbf{e} = e_1 e_2 \ldots e_n$, jehož všechny nenulové složky tvoří část, ležící mezi b po sobě jdoucími znaky. Nebo-li slovo:

$$\mathbf{e} = 000...0 e_i e_{i+1} ... e_{i+b-1} 00...00 . \tag{153}$$

Binární lineární kód objevuje shlukové chyby délky b, jestliže žádné nenulové kódové slovo není shlukem délky b. To znamená, že při vyslání kódového slova **v** a přijetí slova **v** + **e**, kde $\mathbf{e} \neq \mathbf{0}$ je shluk chyb délky b, není **v** + **e** kódové slovo. Při přijetí slova **v** + **e** tedy víme, že došlo k chybě.

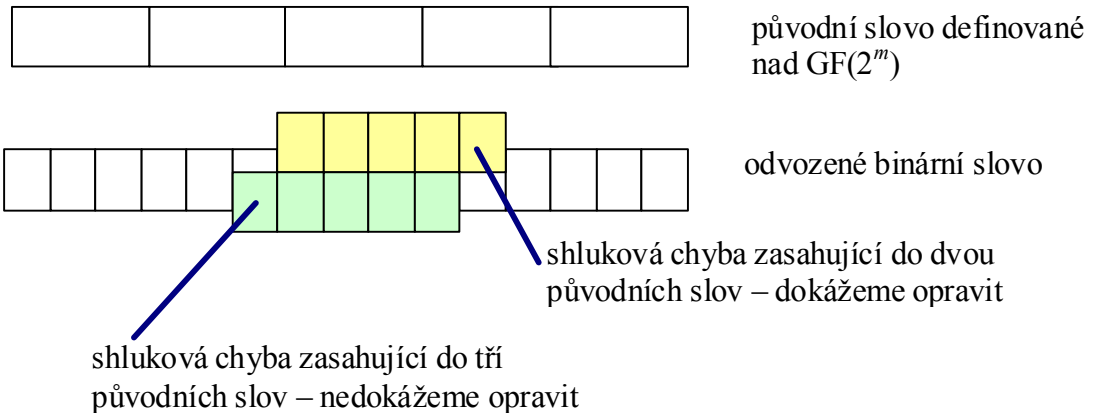
Například cyklický Hammingův (7-4)-kód objevuje shluk chyb délky 3. To je totiž buď slovo:

$$\mathbf{e} = e_0 e_1 e_2 0000 \leftrightarrow e_0 + e_1 z + e_2 z^2 , \tag{154}$$

nebo cyklický posun takového slova. Polynom prvního nebo druhého stupně přitom není kódovým slovem, proto ani jeho cyklický posun není kódovým slovem.

Binární lineární kód opravuje shlukové chyby délky b, jestliže při vyslání slova **v** a přijetí slova **v** + **e**, kde $\mathbf{e} \neq \mathbf{0}$ je shluk chyb délky b, poznáme, že bylo vysláno slovo **v**. Obdobně kód opravuje dva shluky chyb délky b, jestliže při vyslání slova **v** a přijetí slova $\mathbf{v} + \mathbf{e}_1 + \mathbf{e}_2$. kde $\mathbf{e}_1$ a $\mathbf{e}_2$ jsou shlukové chyby délky b poznáme, že bylo vysláno slovo **v**, atd. Binární kódy vytvořené z RS-kódů jsou vhodné pro opravu shlukových chyb.

Nechť K je RS-kód (n, k)-kód pro $n = 2^{m-1}$. Binární kód K^*, který z něj vytvoříme, opravuje shlukové chyby délky $m + 2$. takový shluk totiž poškodí nejvýše dva znaky původního slova (každému znaku původního slova odpovídá $m + 1$ binární znak), viz obr. (9).



Obr. 9. Příklad shlukové chyby na binárním kódovém slově, odvozeném od RS-kódu.

Pokud tedy RS-kód má plánovanou vzdálenost d, pak binární kód K^* opravuje t shlukových chyb délky $m + 2$ kdykoliv je $t < d/4$. Například binární (75, 40)-kód vzniklý z RS-kódu (15, 10)-kódu opravuje shluk délky 6 (protože $m = 4$) a objevuje shlukové chyby délky 17, protože taková shluková chyba způsobí v původním slově poškození pěti znaků a to RS-kód objeví. Tento binární kód má minimální vzdálenost 12, takže opraví 5 chyb a objeví 11 chyb.
Z RS-kódu (15, 7)-kódu vznikne binární (75, 28)-kód. Ten je schopen opravit dva shluky chyb délky 6 (protože $m = 4$ a $d = 6$).

8.2 Využití RS-kódů pro zabezpečení polovodičových pamětí

doprovodná matice

Konstrukci kontrolní matice kódu ukážeme na rozkladu kontrolní matice na tzv. ***doprovodné matice*** **T** nerozložitelného primitivního polynomu

$$g(x) = x^b + g_{b-1}x^{b-1} + ... + g_0 . \tag{155}$$

Matice T je čtvercová matice (b x b) a má tvar:

$$\mathbf{T} = \begin{bmatrix} 0 & 0 & 0 & 0 & ... & ... & ... & g_0 \\ 1 & 0 & 0 & 0 & ... & ... & ... & g_1 \\ 0 & 1 & 0 & 0 & ... & ... & ... & g_2 \\ 0 & 0 & 1 & 0 & ... & ... & ... & g_3 \\ 0 & 0 & 0 & 1 & ... & ... & ... & g_4 \\ ... & ... & ... & ... & 1 & ... & ... & ... \\ ... & ... & ... & ... & ... & ... & ... & ... \\ 0 & 0 & 0 & 0 & ... & ... & 1 & g_{b-1} \end{bmatrix}. \tag{156}$$

Pak mocniny doprovodné matice **T** jsou matice $\mathbf{T}^1, \mathbf{T}^2, \mathbf{T}^3, ..., \mathbf{T}^{2^b-2}, \mathbf{T}^{2^b-1}$. Jestliže má mít RS-kód schopnost opravovat jednu bytovou chybu a detekovat dvě bytové chyby, musí mít kódovou vzdálenost $d = 4$ mezi slovy nebinárního kódu sestrojeného nad Galoisovým tělesem GF(2^b). Kontrolní matice kódu má tvar:

$$\mathbf{H} = \begin{bmatrix} \mathbf{E} & \mathbf{E} & \mathbf{E} & \mathbf{E} & ... & \mathbf{E} \\ \mathbf{E} & \mathbf{T} & \mathbf{T}^2 & \mathbf{T}^3 & \mathbf{...} & \mathbf{T}^{2^b-2} \\ \mathbf{E} & \mathbf{T}^2 & \mathbf{T}^4 & \mathbf{T}^6 & \mathbf{...} & \mathbf{T}^{2(2^b-2)} \end{bmatrix}. \tag{157}$$

Tuto matici je možné prodloužit o soustavu jednotkových matic uspořádaných do konfigurace jednotkové matice (vytvoří jednotkovou matici ve výsledné matici):

$$\mathbf{H} = \begin{bmatrix} \mathbf{E} & \mathbf{E} & \mathbf{E} & \mathbf{E} & ... & \mathbf{E} & \mathbf{E} & \mathbf{0} & \mathbf{0} \\ \mathbf{E} & \mathbf{T} & \mathbf{T}^2 & \mathbf{T}^3 & \mathbf{...} & \mathbf{T}^{2^b-2} & \mathbf{0} & \mathbf{E} & \mathbf{0} \\ \mathbf{E} & \mathbf{T}^2 & \mathbf{T}^4 & \mathbf{T}^6 & \mathbf{...} & \mathbf{T}^{2(2^b-2)} & \mathbf{0} & \mathbf{0} & \mathbf{E} \end{bmatrix}. \tag{158}$$

kód SBC-DBD

Příklad 8.6: RS-kód pro opravu jedné bytové chyby a detekování dvou bytových chyb se nazývá SBC-DBD (Single-Byte Correcting, Double-Byte Detecting). Nerozložitelný polynom třetího stupně nad GF(2) je $g(x) = x^3 + x + 1$. Doprovodná matice **T** má tvar:

$$\mathbf{T} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}. \tag{159}$$

Druhá mocnina doprovodné matice **T** má tvar:

$$\mathbf{T}^2 = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}.\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}. \tag{160}$$

Ostatní potřebné mocniny doprovodné matice **T** mají tvar:

$$\mathbf{T}^3 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}.\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}. \tag{161}$$

$$\mathbf{T}^4 = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}.\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}. \tag{162}$$

$$\mathbf{T}^5 = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}.\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix}. \tag{163}$$

$$\mathbf{T}^6 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix}.\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}. \tag{164}$$

Sedmá mocnina doprovodné matice **T** je shodná s nultou mocninou doprovodné matice, tedy s jednotkovou maticí. Při dalším umocňování se posloupnost matic opakuje. Jedná se tedy o cyklickou multiplikativní grupu konečného tělesa $GF(2^3)$. Kontrolní matice kódu po vyjádření v binární podobě má tvar:

$$\begin{bmatrix}
1&0&0&1&0&0&1&0&0&1&0&0&1&0&0&1&0&0&1&0&0&1&0&0&0&0&0&0&0&0\\
0&1&0&0&1&0&0&1&0&0&1&0&0&1&0&0&1&0&0&1&0&0&1&0&0&0&0&0&0&0\\
0&0&1&0&0&1&0&0&1&0&0&1&0&0&1&0&0&1&0&0&1&0&0&1&0&0&0&0&0&0\\
1&0&0&0&0&1&0&1&0&1&0&1&0&1&1&1&1&1&1&1&0&0&0&0&1&0&0&0&0&0\\
0&1&0&1&0&1&0&1&1&1&1&1&1&1&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0\\
0&0&1&0&1&0&1&0&0&0&1&1&1&1&1&1&1&0&1&0&0&0&0&0&0&0&1&0&0&0\\
1&0&0&0&1&0&0&1&1&1&1&0&0&0&1&1&0&1&1&1&1&0&0&0&0&0&0&1&0&0\\
0&1&0&0&1&1&1&1&0&0&0&1&1&0&1&1&1&1&1&0&0&0&0&0&0&0&0&0&1&0\\
0&0&1&1&0&1&1&1&1&1&0&0&0&1&0&0&1&1&1&1&0&0&0&0&0&0&0&0&0&1
\end{bmatrix}$$

Dekódování RS-kódu podle této kontrolní matice provádíme následovně:

1. Vypočítáme subsyndromy z chybových řádků vektorů jednobytových chyb.
2. Pokud je jeden ze syndromů nenulový a ostatní dva nulové, je chyba v *i*-tém kontrolním bytu.
3. Pokud existuje řešení rovnic subsyndromů, opravíme subsyndromem *i*-tý byte, pokud neexistuje, ohlásíme neopravitelnou chybu.

8.3 Aplikace RS-kódů u optických disků

optická paměť

RS-kódy se nejčastěji používají pro zabezpečování diskových magnetických i optických pamětí. Jako příklad kanálu s výskytem shlukových chyb může nejlépe sloužit optická digitální záznamová technika známá pod názvem kompaktní disk (CD). Paměťové médium pro zvukový záznam na kompaktní disk je plastový kotouč s průměrem 120 mm, tloušťkou 1,2 mm a roztečí záznamových stop 1,6 μm (používají se také CD o průměru 8 cm a CD upravené na velikost kreditní karty). Při přehrávání je informace z disku čtena koherentním optickým paprskem rychlostí 1,25 $m.s^{-1}$. Ve spirální záznamové stopě na disku jsou značky, které jsou nazývány ***jamky*** (pits) a plochá místa mezi jamkami nazývané ***země*** (lands).

jamka

země

RLL-kód

Číslicový zvukový signál je zaznamenán v uspořádání délek jamek a zemí. Symbol „1“ je představován přechodem z jamky na zem nebo naopak, zatímco symbol „0“ je zaznamenán jako setrvání bez přechodu. S ohledem na malé rozměry jamek jsou u kompaktních disků používány RLL-kódy (Runlength-Limited Codes), které kódují vstupní skupiny bitů na výstupní s ohledem na minimalizaci počtu výskytů znaku „1“ a současně tak, aby nevznikly příliš dlouhé posloupnosti znaků „0“. Např (2, 7) RLL-kód je popsán tabulkou:

Vstupní data	(2, 7) RLL kód
11	**1000**
10	**0100**
000	**000100**
010	**100100**
011	**001000**
0011	**00001000**
0010	**00100100**

Pro přenos dat na dlouhé vzdálenosti se používají CD-free RLL kódy, jako např. DC Free (1,7) RLL kód, popsaný tabulkou (x představuje inverzi předchozího bitu):

Vstupní data	DC Free (1,7) RLL kód
00	x01
01	010
10	x00
11 00	010 001
11 01	x00 000
11 10	x00 001
11 11	010 000

Hlavním zdrojem chyb na kompaktních discích jsou nedokonalosti vzniklé při výrobě disků, jako jsou bublinky v plastovém materiálu, nepřesnosti ve tvaru jamek nebo otisky prstů, škrábance, prach, špína a jiná povrchová poškození. Protože každá jamka je asi 0,5 mm široká a dlouhá od 0,9 do 3,3 mm, chyby v kódu se projevují jako shlukové chyby, protože, rušivé vlivy zasahují větší množství informačních bitů. Pro tento paměťový přenosový kanál je tedy nutné použít model kanálu se shlukovými chybami. Je nutné využít kód se schopností opravy shlukových chyb.

Proto jsou zde využívány RS-kódy, které jsou pro tyto účely zvláště vhodné, protože pracují vždy s abecedou se symboly, které jsou ze soustavy vyšší než binární. Pravý i levý kanál zvukového signálu jsou vzorkovány kmitočtem 44,1 kHz a každý vzorek je kvantován šestnáctibitovým slovem. Z toho je zřejmé, že v každém vzorku je kvantováno 32 bitů, neboli čtyři byty. Jednotlivé byty jsou zde nazývány jako ***symbol***, nejsou však shodné se symboly abecedy zdroje. Pro rozpoznání a opravu chyb jsou používány dva RS-kódy. Jedná se ve skutečnosti o dva zkrácené RS-kódy, které jsou získány tak, že některé informační bity jsou položeny rovny nule. Tím se redukují čísla *k* a *n,* ale nemění se nejmenší Hammingova vzdálenost kódu.

symbol

První RS-kód, označovaný C_1 je Reedův-Solomonův (28, 24)-kód. Druhý kód C_2 je Reedův-Solomonův (32, 28)-kód. Abeceda, nad níž jsou oba tyto kódy definovány, je tvořena binárními posloupnostmi o délce osmi bitů. Tím jsou definovány jednotlivé symboly. Vstupní informační posloupnost, přiváděná do kodéru kódu C_1, je tvořena dvaceti čtyřmi symboly. Tato posloupnost je označována jako ***rámec*** (frame) a je kódována do dvaceti osmi symbolů. Symboly, vystupující z kodéru kódu C_1, jsou prokládány, aby se snížil vliv shlukových chyb a aby se chyby rozprostřely na „náhodně rozprostřené“. Takto upravený signál je kódován kodérem C_2 do třiceti dvou symbolů. Na výstupu kodéru kódu C_2 jsou liché symboly každého rámce seřazeny do skupin se sudými symboly příštího rámce. Tím se vytváří nový rámec.

rámec

Na výstupu kodéru kódu C_2 jsou symboly odpovídající šesti vzorkům zvukového signálu zakódovány do třiceti dvou osmibitových symbolů. Navíc je jeden osmibitový symbol přidán. Těchto osm bitů obsahuje informace o uspořádání informace v záznamu. Tím je zkompletován celý rámec, který nyní

EFM-kód

obsahuje třicet tři symboly. Výstupní slova jsou zpracovávána kodérem (14, 8)-kódu (EFM – Eight to Fourteen Modulation, viz obr. 10, který je variantou DC Free RLL kódu), který přiřazuje každému symbolu čtnáctibitovou posloupnost. Dále jsou „přimíchány“ ještě tři bity, což zvyšuje délku posloupnosti na 17, aby bylo zajištěno, že délka kódu bude dostatečná.

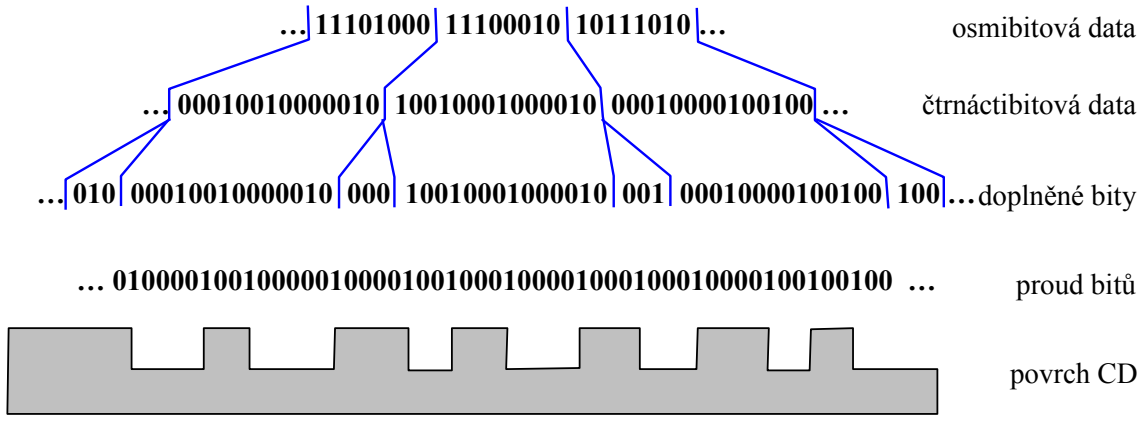
Rámec je dále doplněn dvacetičtyřbitovým synchronizačním vzorkem a třemi dalšími „spojovacími“ bity, aby byla zajištěna dostatečná délka po spojení do rámce. To dává celkový počet kódových bitů na rámec se šesti vzorky 6 x 2 x 16 = 192 bitů na 33 x 17 + 24 + 3 = 588 bitů kanálu. Počet bitů kanálu za jednu sekundu je dán 44 100 : 6 = 4 321 800. Na CD se 67 minutami záznamu je:

$$67 \times 60 \times 4\,321\,800 = 17\,373\,636\,000 \text{ bitů.} \qquad (165)$$

Zpráva je přitom vyjádřena pouze pomocí:

$$67 \times 60 \times 44\,100 \times 32 = 5\,673\,024\,000 \text{ bitů.} \qquad (166)$$

To znamená, že kódová zpráva má přibližně trojnásobnou délku oproti informační zprávě.



Obr. 10. Příklad kódování EFM

Bity, které jsou kódováním přidány, slouží na ochranu informačních bitů před chybami (pomocí RS-kódu) a rovněž zajišťují náležitou délku kódu (EFM kód). Efektivní použití RLL kódu umožňuje zapsat více než 17 miliard kanálových bitů pomocí méně než dvou miliard jamek. Při přehrávání jsou nejprve odděleny synchronizační a spojovací bity a 32 symbolů je zbaveno prokladu. Takto zpracovaný signál vstupuje do dekodéru kódu C_2. Tento kód má minimální kódovou vzdálenost $d = 5$ a je tedy schopen opravovat až dvě chyby.

Dekodér je zkonstruován tak, aby opravoval jenom jednu chybu. Je-li rozpoznána jedna chyba, je opravena, je-li chyb více, je všech 28 symbolů označeno jako nespolehlivá informace. Potom, co je odstraněn proklad, jsou symboly dekódovány dekodérem kódu C_1. Kód C_1 má rovněž nejmenší kódovou vzdálenost $d = 5$. Dekodér je opět navržen tak, aby opravoval jednu chybu a rozpoznával dvě nebo více chyb. Na výstupu druhého dekodéru jsou

symboly, které odpovídají polohou nespolehlivé informaci, nahrazeny hodnotou, která je interpolací sousedních hodnot. Použitím této složité kódovací a dekódovací techniky spolu se zpracováním signálu mohou být překlenuty chyby až do počtu 12 000 chybějících informačních bitů, což odpovídá rozměru 7,5 mm^2 na povrchu kompaktního disku.

Rea1izace kódového systému pro zabezpečení informací na optickém kompaktním disku je vzhledem k hromadnosti realizací řešeno jako specializovaný obvod, který má podobnou architekturu jako signálový procesor. Specializovaný obvod má vysoce výkonnou paralelní výpočetní jednotku. Podpůrné obvody, které dále zvyšují výkon obvodu, jsou zaměřeny na nezávislá komunikační rozhraní, která usnadňují začlenění kódového systému do aplikačního zapojení.

Kontrolní otázky:

30. Jak jsou definovány RS-kódy?
31. Jak se používají RS-kódy?
32. Jak je zabezpečena optická paměť?

Shrnutí obsahu kapitoly

V této kapitole jsme se seznámili s rozšířením třídy BCH-kódů pro odhalování a opravu shlukových chyb – Reedovy-Solomonovy kódy. Umíme sestavit RS-kód i popsat algoritmus jeho dekódování včetně opravy chyb. Víme, jak se RS-kódy používají u optických pamětí.

9 Šifrování dat

Cíl:

Cílem celé kapitoly je seznámení se základními pojmy z oblasti kryptografické ochrany dat, je vysvětlen obsah kryptografie a jejích částí, kryptografie a kryptoanalýzy. Jsou vysvětleny základní pojmy a definovány základní principy kryptografických systémů. Na závěr je ukázán příklad utajeného přenosu spadajícího do oblasti Steganografie.

Po jejím prostudování byste měli být schopni:

- Definovat základní pojmy z oblasti kryptografie.
- Klasifikovat a popsat jednotlivé typy šifrovacích systémů.
- Popsat základní charakteristiky zdroje zpráv využívané při kryptoanalýze.
- Ukázat příklady utajeného přenosu zpráv.

Klíčová slova této kapitoly:

Kryptologie, kryptografie, kryptoanalýza, utajený přenosový kanál, Steganografie, veřejný přenosový kanál, systém s tajným klíčem, systém s veřejným klíčem, otevřený text, transpoziční systém, transkripční systém, proudový systém, blokový systém, kódování, kód, abeceda textu, kryptografický systém, prosté zobrazení, monoalfabetický systém, polyalfabetický systém, bezpaměťový zdroj zpráv, digram, trigram, index koincidence, utajený přenos, Baconova šifra.

Doba potřebná ke studiu: 2 hodiny

Průvodce studiem

Studium této kapitoly je nutné k pochopení postupů kryptografické ochrany dat. kapitola prezentuje a vysvětluje základní pojmy kryptografie a kryptoanalýzy. Jsou prezentovány základní principy kryptoanalýzy vycházející z analýzy zdroje zpráv a jeho specifických vlastností. na závěr kapitoly je ukázán příklad utajeného přenosu zpráv pomocí Baconova systému.
Na studium této části si vyhraďte dvě hodiny.

9.1 Ochrana informace

kryptologie

Oborem ***kryptologie*** je utajování zpráv přenášených od zdroje k příjemci pomocí informačního kanálu, který může být obsazen (odposloucháván) narušitelem. Současně také ochrana před mylnými zprávami vyslanými narušitelem. Tím se zásadně liší od oboru kódování. I když jsou si svými prostředky a postupy často velmi blízké, jak uvidíme dále.

utajený přenosový kanál

Prvopočátky ochrany informace začínají spolu s přenosem zpráv. Zpočátku byly snahy o utajení přenášené zprávy směřovány do ***utajení přenosového kanálu***. Historicky nejstarší dochovaný popis takového systému pochází z antického Řecka. Zpráva byla napsána na oholenou hlavu otroka. Po narostení vlasů byl otrok odeslán. Přenos jistě nevynikal rychlostí. Navíc se záhy zjistilo, že prakticky neexistuje dokonale utajený přenosový kanál. Každý kurýr mohl být přepaden, či jednodušeji koupen. Přesto se snahy o budování tajných přenosových kanálů nebo ukrývání zpráv v jinak nezávadném textu dále využívají, a různé tajné inkousty, vypíchané tečky pod písmeny v knize apod. stále nejsou minulostí. Zabývá se jimi obor ***Steganografie***, v současnosti je rozvíjeno např. ukrývání zpráv v binárních datech obrázků, rozvíjejí se snahy ukrývání zpráv v DNA řetězcích apod.

Steganografie

veřejný přenosový kanál

Vývoj tedy dospěl k budování šifrovacích postupů, které zajišťovaly bezpečnost informace i přesto, že se k nim mohla dostat nepovolaná osoba. Jejich nutnost postupně vzrůstala s rozvojem veřejných datových sítí, počínaje poštovními službami, přes telegrafní a bezdrátové spojení a konče počítačovými sítěmi jako Internet. Jejich nasazení vycházelo z použití ***veřejného přenosového kanálu***, který může být odposloucháván.

systém s tajným klíčem

Během historického vývoje vznikla celá řada šifrovacích postupů založených na utajeném způsobu šifrování, případně další informaci potřebné pro správné použití této metody, kterou nazýváme klíčem. Tyto systémy se tedy souhrnně nazývají „***Systémy s tajným klíčem***“. Pokud došlo k prozrazení klíče, byla zpráva dešifrována. Jejich použitelnost je dána nemožností zjistit klíč přímo ze šifrované zprávy. Proto se budeme věnovat také postupům, jak tento klíč ze šifrované zprávy zjistit.

systém s veřejným klíčem

V současné době se prosazují v ochraně informace spíše „***Systémy s veřejným klíčem***“. Tedy systémy, ve kterých je nejen šifrovací postup, ale také klíč k zašifrování zprávy běžně znám. Jsou založeny na skutečnosti, že z dostupné informace nelze získat v rozumném čase klíč pro dešifrování zprávy.

otevřený text

Podle způsobu přístupu k šifrování ***otevřeného textu*** (původní zprávy, otevřené zprávy, zprávy v otevřené řeči) můžeme rozdělit šifry do následujících skupin:

- ***Utajený přenos*** – sem zařazujeme všechny snahy o utajený přenos zpráv, tedy postupy budující utajené přenosové kanály. Mohou být tvořeny prověřenými osobami, použitím tajného inkoustu či jinými způsoby tajnopisu – Steganografie.
- ***Transpoziční systémy*** – provádějí určenou manipulaci s jednotlivými znaky, vyměňují jejich pozici v textu apod., aby zpráva byla nečitelná.

transpoziční systém

I přes jejich dřívější značné rozšíření jsou snáze dešifrovatelné, záhy se používaly pouze jako doplněk složitějšího šifrovacího postupu. Jejich poslední významné strategické využití je známo z období první světové války.

transkripční systém

- ***Transkripční systémy*** – ponechávají původní znaky na stejných místech, ale zaměňují je za znaky jiné. Výraznou výhodou je možnost šifrovat jakkoliv dlouhou zprávu. Později se začaly rozdělovat na systémy:

proudový systém

  - ***proudové*** – kdy s každým znakem provedeme jednu šifrovací operaci

blokový systém

  - ***blokové*** – kdy s blokem textu provádíme šifrovací operaci opakovaně v několika cyklech (rundách), čímž dosahujeme vyšší kvality systému.

Zakončeme tuto úvodní kapitolu vymezením působnosti kódování a šifrování. Řada metod šifrování nahrazuje znaky ***otevřeného textu*** (otevřené zprávy, zprávy v otevřené řeči) jinými znaky. Tím v zásadě nedělá nic jiného, než ***kódování***. Skutečně je tomu tak. Jediný rozdíl je v tom, že použité přiřazení znaků (tzv. kódová kniha) není veřejně známo. Tedy použitý ***kód*** je tajný. Jinak musí kryptografické systémy splňovat podobné vlastnosti jako kódy, musí tedy jít o prostá přiřazení apod.

kódování

kód

9.2 Statistické vlastnosti zdroje zpráv

9.2.1 Abeceda zdroje zpráv

Obvykle se pod pojmem šifrování chápe postup, který přiřazuje znakům původní zprávy znaky ze stejného souboru znaků. Použitý soubor znaků, s jejichž pomocí je zpráva sestavena, se nazývá ***abeceda textu***. Tento soubor znaků označujeme Z_N, kde N – počet prvků souboru.

abeceda textu

Hlavním atributem abecedy textu je tedy počet jejích prvků. Nejčastěji budeme pracovat s následujícími abecedami:

Z_{26} –Telegrafní abeceda obsahující 26 znaků:

ABCDEFGHIJKLMNOPQRSTUVWXYZ,

Z_{27} – Telegrafní abeceda s mezerou,

Z_{41} – Česká abeceda velkých písmen:

AÁBCČDĎEÉĚFGHIÍJKLMNŇOÓPQRŘSŠTŤUÚŮVWXYÝZŽ,

Z_{42} – Česká abeceda velkých písmen s mezerou,

Z_{128} – Abeceda kódu ASCII znaků (7 bitů),

Z_{256} – Abeceda rozšířeného kódu ASCII-2 (8 bitů).

Pro zjednodušení operací s abecedou budeme znaky abecedy označovat čísly 0, 1, 2, ..., n – 1 v pořadí jejich výskytu v abecedě.

Každá šifra pak představuje kryptografickou transformaci T

$$T: Z_{N,n} \rightarrow Z_{N,n}, \tag{71}$$

kde je N – počet znaků abecedy,
n – počet znaků zprávy.

Tato transformace musí být prostým zobrazením (pro $x \neq y$ platí $f(x) \neq f(y)$).

Jako ***kryptografický systém*** (způsob šifrování) chápeme parametrický systém:

kryptografický systém

$$T^k = \{ T_k: k \in K \}, \tag{72}$$

kde je k – klíč, prvek množiny K,
K – prostor klíčů.

Základní principy kryptografických systémů pak jsou:

1. Text původní a šifrované zprávy je psán nad toutéž abecedou.
2. Délky původního a šifrovaného textu jsou stejné.
3. Každému otevřenému textu je přiřazen jediný šifrovaný text, neboť kryptografická transformace je zobrazení.
4. Každý text nad abecedou Z_N umíme zašifrovat právě jedním způsobem, neboť kryptografická transformace je ***prosté zobrazení***.

prosté zobrazení

Při dešifrování zpráv bylo zjištěno, že se různé znaky abecedy vyskytují v textu různě často. Na základě tohoto poznání pak rozdělujeme systémy s tajným klíčem do dvou skupin:

- Kryptografický systém, který každému znaku otevřené zprávy přiřazuje vždy týž znak šifrované zprávy, se nazývá ***monoalfabetický*** systém.

monoalfabetický systém

- Pokud stejný znak otevřené zprávy může být nahrazen různými znaky výsledné zprávy, hovoříme o ***polyalfabetickém*** systému.

polyalfabetický systém

9.2.2 Zdroj zpráv

Základem pro analýzu a rozluštění šifrované zprávy je analýza zdroje zpráv. Již bylo zmíněno, že jednotlivé znaky se ve zprávě vyskytují různě často. Obvykle přitom předpokládáme, že zdroj generuje jednotlivé znaky abecedy nezávisle na předchozích. (U jazyka to není pravda, samohláska se vyskytuje mnohem častěji za souhláskou, než za samohláskou apod.).

Budeme tedy pracovat s tzv. ***bezpaměťovým zdrojem zpráv***.

bezpaměťový zdroj zpráv

Pravděpodobnost, že byla vyslána právě zpráva $\{x_1, x_2, ..., x_n\}$ je rovna:

$$P(\{x_1, x_2, ..., x_n\}) = P(x_1).P(x_2)... P(x_n), \tag{73}$$

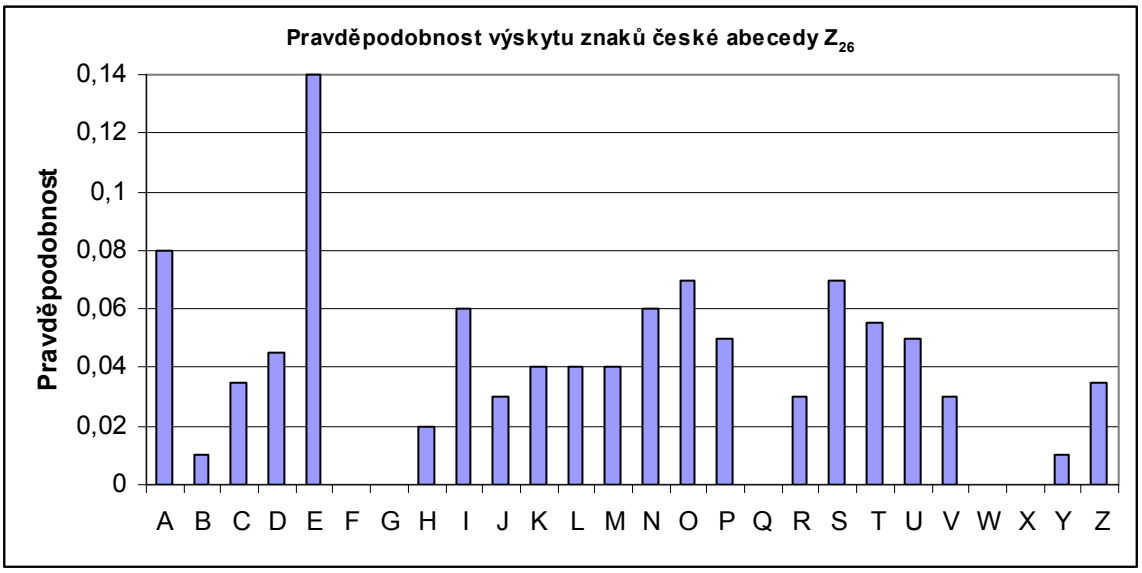
kde je $P(x_i)$ – pravděpodobnost výskytu znaku x_i v textu.

Přitom víme, že jednotlivé znaky se nevyskytují stejně často. Jejich pravděpodobnosti[#] byly pro každý jazyk určeny rozborem řady textů různých autorů i obsahu a bývají k dispozici. Musíme přitom mít na paměti, že tyto pravděpodobnosti nemusí u krátkých textů platit bez výhrad. V tab. 1 jsou uvedeny pravděpodobnosti výskytů znaků českých textů pro abecedu Z_{26}, tab. 2 pak ukazuje grafické vyjádření pravděpodobnosti výskytů jednotlivých znaků českého textu psaného v abecedě Z_{26}. V tab. 3 je uvedena pravděpodobnost výskytu znaků českých textů pro abecedu Z_{42} [19]. Nutno přiznat, že údaje se v různých pramenech i zásadně liší, viz srovnání s tab. 4 [79].

Tab. 1. Pravděpodobnosti výskytu jednotlivých znaků české abecedy Z_{26}

A	**0,08**	**J**	**0,03**	**S**	**0,07**
B	**0,01**	**K**	**0,04**	**T**	**0,055**
C	**0,035**	**L**	**0,04**	**U**	**0,05**
D	**0,045**	**M**	**0,04**	**V**	**0,03**
E	**0,14**	**N**	**0,06**	**W**	**0,00**
F	**0,00**	**O**	**0,07**	**X**	**0,00**
G	**0,00**	**P**	**0,05**	**Y**	**0,01**
H	**0,02**	**Q**	**0,00**	**Z**	**0,035**
I	**0,06**	**R**	**0,03**		

Tab. 2. Histogram českého textu v Z_{26}



[#] Pojem pravděpodobnost je chápán jako teoretická (přesná) hodnota četnosti (frekvence) výskytu znaků. Frekvenci výskytu znaků obvykle určujeme vyšetřením zvolené množiny textů.

Tab. 3. Pravděpodobnosti výskytu jednotlivých znaků české abecedy Z_{42}

A	**0,054**	**F**	**0,002**	**Ň**	**0,015**	**Ť**	**0,007**	**Ž**	**0,009**
Á	**0,021**	**G**	**0,002**	**O**	**0,068**	**U**	**0,030**	**–**	**0,163**
B	**0,014**	**H**	**0,020**	**Ó**	**0,000**	**Ú**	**0,003**		
C	**0,019**	**I**	**0,034**	**P**	**0,027**	**Ů**	**0,002**		
Č	**0,008**	**Í**	**0,025**	**Q**	**0,000**	**V**	**0,039**		
D	**0,026**	**J**	**0,022**	**R**	**0,029**	**W**	**0,000**		
Ď	**0,005**	**K**	**0,033**	**Ř**	**0,009**	**X**	**0,001**		
E	**0,073**	**L**	**0,034**	**S**	**0,040**	**Y**	**0,016**		
É	**0,010**	**M**	**0,029**	**Š**	**0,008**	**Ý**	**0,008**		
Ě	**0,007**	**N**	**0,040**	**T**	**0,039**	**Z**	**0,019**		

Tab. 4. Pořadí četnosti výskytu jednotlivých znaků české abecedy Z_{41}

pořadí	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
celkové	e	o	a	i	t	s	n	l	i	k	v	m	r	p	j	u	d	á	ň	z
začátek slova	s	p	n	v	j	t	z	m	a	k	d	o	b	f	r	h	u	l	ž	c
konec slova	e	i	a	í	o	m	u	é	t	k	ch	ou	á	l	s	š	n	f	v	ú

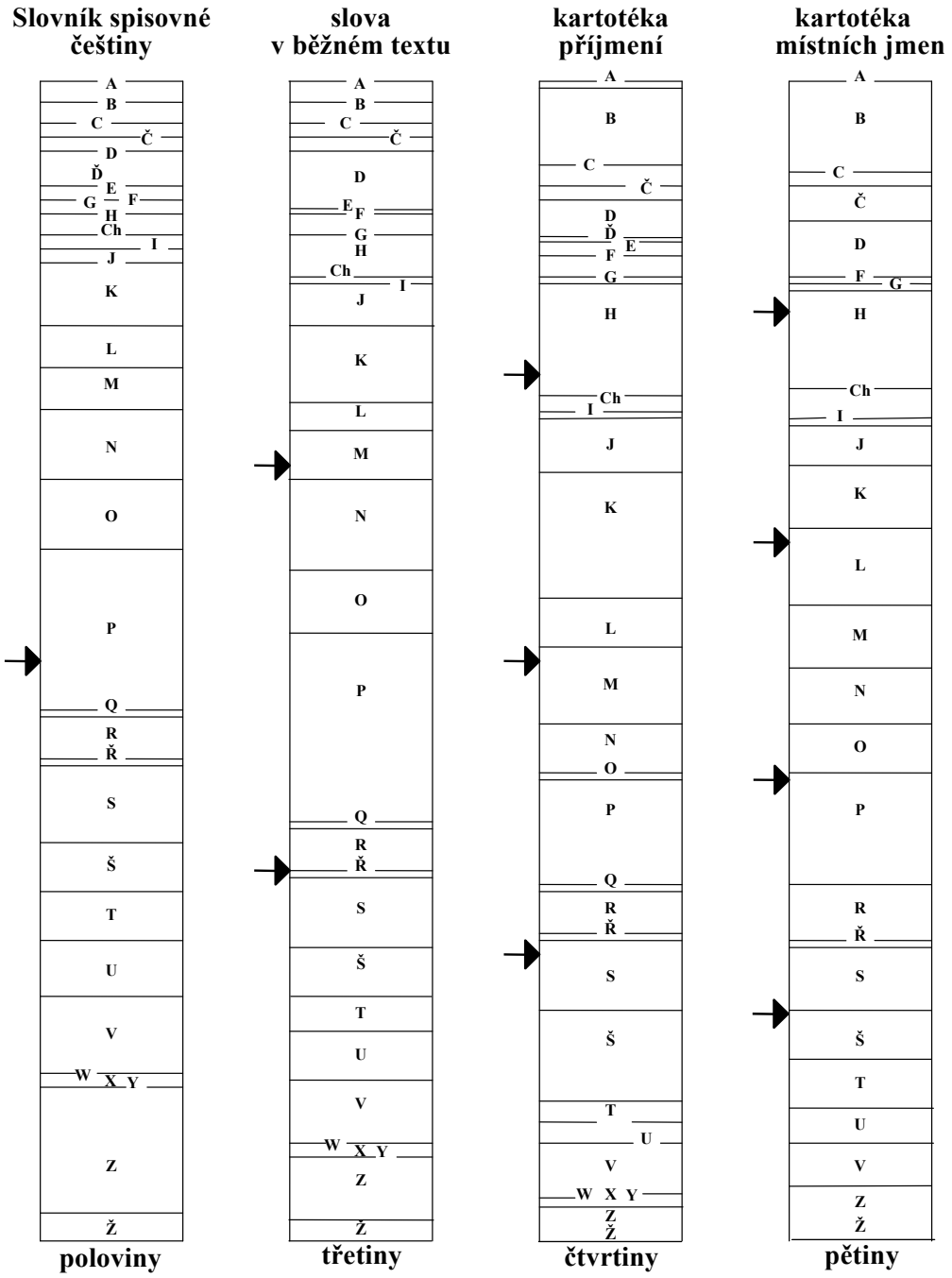
Podobně se zjišťují pravděpodobnosti výskytu dvojic znaků (***bigramů***), trojic znaků (***trigramů***), nebo celých slov, viz např. [29]. Se znalostí těchto pravděpodobností můžeme přistoupit k dešifrování zpráv, které jsme zachytili.

bigram

trigram

Znalost frekvencí (četností) výskytů jednotlivých znaků v textu má samozřejmě hlubší význam. Používají ji např. výrobci písmen pro zhotovování sad písmen (Propisot), dříve ji používali tiskaři, pokud se ještě texty sázely ručně, aby si připravili vhodnou zásobu liter apod.

Znalost frekvence počátečních písmen je důležitá, pokud vytváříme nějakou kartotéku a potřebujeme ji dělit na části. Vhodné rozdělení jednotlivých částí je zřejmé z obr. 11. [79].



Obr. 11. Frekvence počátečních písmen českých slov

index koincidence

Při dešifrování zprávy je prvním krokem zjištění, zda byl použit transpoziční nebo transkripční systém, dále zda byl použit monoalfabetický nebo polyalfabetický systém šifer (případně také v jakém jazyce byla zpráva psána, pokud to netušíme). K tomu slouží ***index koincidence*** (viz dále).

Vycházíme ze skutečnosti, že známe rozložení pravděpodobnosti výskytů jednotlivých znaků v otevřené řeči. Pokud bychom použili dokonalý polyalfabetický systém, pak pravděpodobnost výskytu každého znaku v šifrované zprávě bude:

$$\frac{1}{N}, \quad pro \quad Z_{26} \Rightarrow \frac{1}{26} \doteq 0{,}03846. \tag{74}$$

Určeme součet kvadratických odchylek skutečných pravděpodobností výskytu znaků pro zprávu C psanou v Z_N. Jednotlivé pravděpodobnosti určíme jako podíl počtu výskytů daného znaku ku počtu znaků celé zprávy.

$$V(C) = \sum_{i=0}^{N-1} \left(P(x_i) - \frac{1}{N} \right)^2, \tag{75}$$

Po úpravě rozvinutím závorky dostáváme:

$$V(C) = \sum_{i=0}^{N-1} P^2(x_i) - \frac{2}{N} \cdot \sum_{i=0}^{N-1} P(x_i) + \sum_{i=0}^{N-1} \left(\frac{1}{N} \right)^2 = \sum_{i=0}^{N-1} P^2(x_i) - \frac{2}{N} + N \cdot \frac{1}{N^2},$$

$$V(C) = \sum_{i=0}^{N-1} P^2(x_i) - \frac{1}{N}. \tag{76}$$

Pro abecedu Z_{42} (česká s mezerou) je

$$V(C) \doteq 0{,}057016 - \frac{1}{42} \doteq 0{,}033206$$

pro monoalfabetickou šifru. Pro dokonalou polyalfabetickou šifru je $V(C) = 0$.
Pro zjednodušení se obvykle pracuje jen se součtem kvadrátů pravděpodobností:

$$V(C) = \sum_{i=0}^{N-1} P^2(x_i). \tag{77}$$

Ten vyjadřuje pravděpodobnost, že dva znaky jdoucí v textu za sebou jsou stejné. My máme k dispozici jen šifrovaný text a hodnotu $V(C)$ z něj jen odhadujeme. Vyjdeme tedy z počtu výskytů znaků v textu. Počet dvojic stejného znaku, které můžeme vytvořit, je dán následujícím vztahem:

$$\binom{f(x_i)}{2} = \frac{1}{2} \cdot f(x_i) \cdot [f(x_i) - 1], \tag{78}$$

kde je $f(x_i)$ – počet znaků x_i v textu.

Počet všech dvojic znaků, které můžeme vytvořit je:

$$\binom{\sum_{i=0}^{N-1} f(x_i)}{2} = \frac{1}{2} \cdot \sum_{i=0}^{N-1} f(x_i) \cdot \left[\sum_{i=0}^{N-1} f(x_i) - 1 \right]. \tag{79}$$

Pak index koincidence $I(C)$ „aproximující“ součet druhých mocnin pravděpodobností výskytů znaků je dán vztahem

$$I(C) = \frac{\sum_{i=0}^{N-1} \binom{f(x_i)}{2}}{\binom{\sum_{i=0}^{N-1} f(x_i)}{2}} = \frac{\sum_{i=0}^{N-1} \{ f(x_i) \cdot [f(x_i) - 1] \}}{\sum_{i=0}^{N-1} f(x_i) \cdot \left[\sum_{i=0}^{N-1} f(x_i) - 1 \right]}. \tag{80}$$

Pokud se $I(C)$ blíží hodnotě $\sum_{i=0}^{N-1} P^2(x_i)$, jde o monoalfabetickou šifru, pokud se blíží $\frac{1}{N}$, pak jde o šifru polyalfabetickou.

9.3 Utajený přenos

utajený přenos

Pro zajímavost uveďme alespoň jednu ukázku tajného kódu zapisovaného do jinak zcela nezávadného textu. Používal ho Lord Francis Bacon (1561-1626). V nezávadném textu používal dva typy písma, třeba stojaté a ležaté znaky. Každé písmeno tajné zprávy nejprve převedl na pětici znaků majících jen jednu ze dvou hodnot, znázorněme je třeba pomocí znaků 0 a 1. Nula znamená ležaté písmeno, jednička stojaté písmeno. Podle toho pak zapsal nezávadný text.

Baconova šifra

Třeba v textu:

Mi*lý* **p***říte***l**i, *d***o***r***uč***i***te***l** *t***o***hoto d***opi***s***u je** *m***i zvlášť milý**.

najdeme tuto utajenou zprávu:

11001 00000 10101 01001 01000 00111 01110 11111 11111 1

která dává skrytý text: Zavři ho!

Příklad přiřazení znaků Baconovy šifry:

a	00000	b	00001	c	00010	d	00011	e	00100	f	00101	g	00110
h	00111												
i	01000	j	01001	k	01010	l	01011	m	01100	n	01101	o	01110
p	01111												
q	10000	r	10001	s	10010	t	10011	u	10100	v	10101	w	10110
x	10111												
y	11000	z	11001.										

Celkem takto lze vyjádřit $2^5 = 32$ různé znaky.

Kontrolní otázky:

33. Definujte obor kryptologie, kryptografie a kryptoanalýzy?
34. Jaké vlastnosti zdroje zpráv používáme při kryptoanalýze?
35. Jak je definován index koincidence a k čemu slouží?

Korespondenční úkol:

4. Vytvořte příklad steganografického systému a ukažte na hodném příkladě jak funguje.

Shrnutí obsahu kapitoly

V této kapitole jsme se seznámili se základními pojmy z oblasti kryptografické ochrany dat, klasifikovali jednotlivé typy kryptografických systémů. Současně jsme se seznámili s charakteristikami zdroje zpráv, využívaných při kryptoanalýze. Na závěr byly prezentovány principy utajeného přenosu zpráv a ukázán příklad takového systému.

10 Transpoziční systémy

Cíl:

Cílem celé kapitoly je seznámení s principem transpozičních kryptografických systémů. Jsou prezentovány příklady transpozičních systémů od nejstarších jako jednoduchá transpozice po nejkomplikovanější, používané ještě za první světové války, mezi které patří otočná šifrovací mřížka. Přestože z dnešního pohledu jsou již překonány, je vhodné pochopit, proč tomu tak je.

Po jejím prostudování byste měli být schopni:

- Popsat příklady různých transpozičních kryptografických systémů.
- Popsat metody k narušení transpozičních kryptografických systémů.
- Vysvětlit proč nejsou transpoziční systémy z dnešního pohledu nepoužitelné.

Klíčová slova této kapitoly:

Transpoziční systém, Scytala, klíč, jednoduchá transpozice, heslo, dvojitá transpozice, šifrovací mřížka, Cardanova mřížka, klamač, otočná mřížka, Fleissnerova mřížka, mohutnost množiny klíčů.

Doba potřebná ke studiu: 4 hodiny

Průvodce studiem

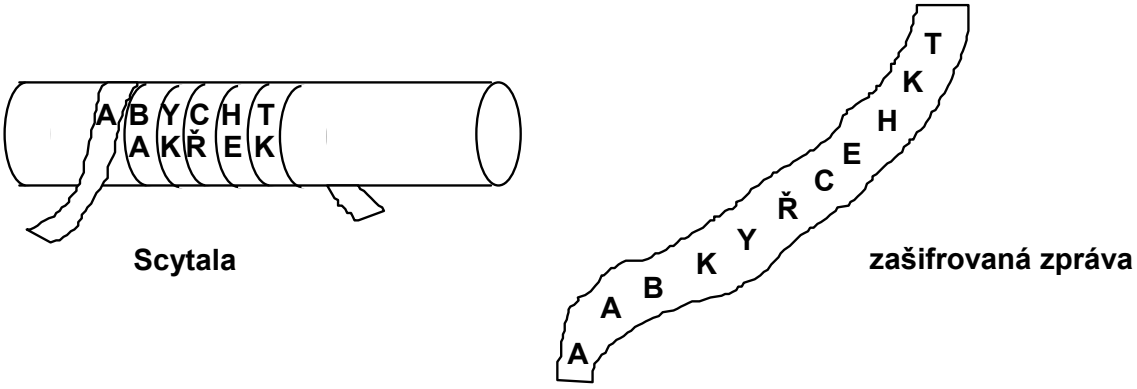
Studium této kapitoly je nutné k pochopení funkce transpozičních kryptografických systémů. Je prezentován historický průřez jejich vývojem a dokumentovány příklady různých kryptografických systémů včetně jejich slabých stránek, které způsobily, že v dnešní době již nejsou prakticky využitelné.
Na studium této části si vyhraďte čtyři hodiny. Vyzkoušejte si, jak fungují různé transpoziční systémy.

Transpoziční kryptografické systémy jsou historicky starší. Nejstarší dochovaný systém pochází z antického Řecka, jmenovitě ze Sparty, kde ho popsal řecký historik Plutarchos. Je to současně nejstarší dochovaný systém využívající šifrovací pomůcku pod názvem ***Scytala***. Na dřevěnou tyč navineme proužek pergamenu a na něj zprávu napíšeme. Bez znalosti správného průměru použité tyče je zpráva nečitelná (průměr tyče je tedy vlastně ***klíčem*** k dešifrování). Tedy tehdy byla.

transpoziční systém

Scytala

klíč



Obr. 12. Šifrovací pomůcka Scytala

Pak se vývoj šifrovacích systémů dostává na dlouhou dobu do pozadí zájmu. Opět se objevuje až po skončení středověku, spolu s rozvojem všech vědních oborů. Autorem první knihy o šifrování je benediktinský opat ze Spanheimu Johannes Trittheim (1462-1518). Věnoval se v ní právě transpozičním systémům. Vzbudila však nevoli v panovnických kruzích, takže byl autor označen za čarodějníka.

10.1 Jednoduchá transpozice

Mezi první novověké transpoziční systémy patří ***jednoduchá transpozice***, kterou používal kardinál Richelieu (✞ 1614). K šifrování použijeme ***heslo*** (klíč) tvořené jedním slovem nebo skupinou slov zvolené abecedy. To nám určí pořadí, ve kterém bereme písmena otevřeného textu. Ten samozřejmě musíme rozdělit do bloků stejné délky jako je heslo. Vidíme tu typický požadavek transpozičních systémů, skutečnost, že zpráva musí být prodloužena na požadovanou délku.

jednoduchá transpozice

heslo

heslo	**Z**	**L**	**A**	**T**	**A**	**B**	**R**	**A**	**N**	**A**
pořadí	**10**	**6**	**1**	**9**	**2**	**5**	**8**	**3**	**7**	**4**
otevřená zpráva	K	L	O	K	A	N	P	R	I	L
	E	T	I	V	P	R	O	S	I	N
	C	I	X	X	X	X	X	X	X	X

Šifrovaná zpráva se obvykle rozděluje do pětipísmenných skupin a zní:
OARLN LIPKK IPSNR TIRVE XXXXX IXXXC

V tomto směru byly vedeny další práce, podobně pracoval plukovník Roche, či italský fyzik Gian Babtista della Porta (1563), viz [29]. Postup šifrování a dešifrování je poněkud zdlouhavý, proto se později přešlo na jiný postup,

používaný dodnes. Zpráva byla opět zapsána do podobné tabulky, ale šifrovaná zpráva se četla po sloupcích v pořadí určeném čísly sloupců. Odpadla tím i nutnost zarovnávat text na násobek délky klíče:

heslo	**Z**	**L**	**A**	**T**	**A**	**B**	**R**	**A**	**N**	**A**
pořadí	**10**	**6**	**1**	**9**	**2**	**5**	**8**	**3**	**7**	**4**
otevřená zpráva	K	L	O	K	A	N	P	R	I	L
	E	T	I	V	P	R	O	S	I	N
	C	I								

Šifrovaný text zní:
OIAPR SLNNR LTIII POKVK EC

Doplňování textu na násobek délky klíče výrazně zjednodušuje narušení systému. Délky hesla se u těchto systémů pohybují kolem 10-30 znaků, což dává příliš malé množství kombinací, než aby nemohly být všechny vyzkoušeny. Proto se systém postupně komplikoval, např. ***dvojitou transpozicí***. Text po první transpozici byl pomocí druhého hesla transponován podruhé.

dvojitá transpozice

10.2 Šifrovací mřížka

šifrovací mřížka

Cardanova mřížka

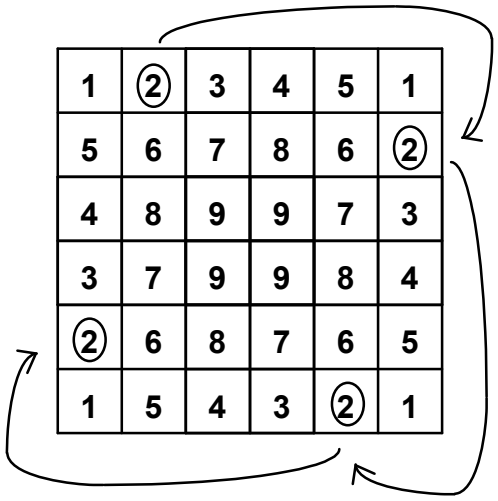
klamač

Jiný postup zvolil ve svém systému renesanční učenec Jeroným Cardan v 16. stol. Navrhl použití ***šifrovací mřížky*** s několika otvory. Do nich se po řádcích zapíše text zprávy, doplní se na volných místech nezávadným textem a přepíše po sloupcích. Znaky doplňkového textu se nazývají ***klamače***, používají se i v jiných systémech ke zmatení luštitele. V praktickém použití byly klamače přímo natištěny na použité mřížce. Šifrovací mřížka se ukázala jako velmi silný nástroj, proto byla dále rozvíjena až do 18. a 19. století. Přispěli k tomu zejména Hindenburg de Presse, němečtí diplomaté Kluber a Martens a zejména rakouský plukovník Fleissner von Wostrowitz ve své knize z roku 1881. Ten navrhl mřížku otočnou, takže text byl jednak lépe „rozházen“ a dále odpadla nutnost přidávání klamačů.

Fleissnerova mřížka

otočná mřížka

Fleissnerova mřížka (otočná mřížka) je čtvercová, má vždy sudý počet znaků. Přiložíme ji ke zprávě a po řádcích přečteme text, pak mřížku pootočíme a pokračujeme ve čtení. Druhou možností je napsat otevřený text do čtvercového prostoru a po přiložení mřížky číst jednotlivé znaky šifrované zprávy. To nám spolu se dvěma směry otáčení (doleva a doprava) dává čtyři možné postupy šifrování. Aby mřížka fungovala na všechny čtyři polohy, musíme si uvědomit, jak se bude otvor při otáčení stěhovat. Víme, že zabere postupně čtyři různá místa, z těchto míst může být v mřížce vyděrováno jen jedno. Celá situace je znázorněna na obr. 13. Políčka, na která se dostane stejný otvor, jsou očíslována vždy stejným číslem. Pohyb otvoru je znázorněn na políčku s číslem 2.



Obr. 13. Princip Fleissnerovy otočné mřížky

Dostaneme-li samostatný text bez šifrovací mřížky, pak je poměrně těžké ho rozluštit. Proč tomu tak je, si uvědomíme, když spočítáme, kolik různých šifrovacích mřížek (různých klíčů) je možno sestavit. Počet možných klíčů (***mohutnost množiny klíčů***) je jedním ze základních kritérií pro hodnocení kvality kryptografického systému.

mohutnost množiny klíčů

U nejmenší mřížky s rozměrem 4 x 4 políčka vystřihujeme celkem čtyři otvory (při čtyřech polohách tak zaplníme celou mřížku). Každý otvor můžeme vystřihnout na jednom ze čtyř míst. Máme tedy celkem 4 x 4 x 4 x 4 možností, to je celkem $4^4 = 256$ možností.

Pro větší mřížky tento počet prudce roste:

mřížka 6 x 6 dává	4^9 =	262 144 možností,
mřížka 8 x 8 dává	4^{16} =	4 294 967 296 možností.
mřížka 10 x 10 dává	4^{25} =	1 125 899 906 843 000 možností.

To jsou již těžko představitelná čísla. To byl, zřejmě spolu s jednoduchostí ovládání mřížky, hlavní důvod, proč se šifrovací mřížka používala ještě v první světové válce (tedy na začátku 20. století) jako polní vojenská šifra v německé armádě.

V současnosti se transpoziční systémy pro zabezpečení dat samostatně nepoužívají. Vznikaly v době ručního zpracování a pro zpracování počítačem se příliš nehodí, navíc jsou dnes běžně luštěny i amatérskými prostředky. Proto se jimi také nebudeme dále zabývat. Bližší informace je možno najít v [16, 29, 49] apod.

Kontrolní otázky:

36. Jaký je základní princip činnosti transpozičních kryptografických systémů?
37. Jak funguje dvojitá transpozice, mohou být délky použitých hesel libovolné?
38. Jaký význam má mohutnost množiny klíčů?

Korespondenční úkol:

5. Ukažte na vhodném příkladě použití dvojité transpozice. Diskutujte možné problémy při volbě použitých hesel.

Shrnutí obsahu kapitoly

V této kapitole jsme se seznámili s typickými příklady transpozičních kryptografických systémů od historicky nejstarších po nejkomplikovanější. Současně jsme si ukázali význam klíče kryptografického systému a zejména význam mohutnosti množiny klíčů.

11 Transkripční systémy

Cíl:

Cílem celé kapitoly je seznámení s principem transkripčních kryptografických systémů. Jsou prezentovány příklady transkripčních systémů od nejstarších jako je Cézarovská šifra, po nejkomplikovanější, které jsou použitelné i dnes. Současně je prezentován matematický aparát potřebný pro popis systému i jeho automatizovanou realizaci.

Po jejím prostudování byste měli být schopni:

- Popsat příklady různých transkripčních kryptografických systémů.
- Popsat metody k narušení transkripčních kryptografických systémů.
- Vysvětlit za jakých podmínek jsou transkripční systémy použitelné i v dnešní době.

Klíčová slova této kapitoly:

Transkripční systém, homofonie, monoalfabetický systém, Cézarovská šifra, kryptografická transformace, afinní šifra, multiplikativní šifra, kongruence, obecná monoalfabetická šifra, klíčová fráze, heslo, heslem míchaná abeceda, nomenklátor, autoklíč, autokláv, klíč, kotouče různého průměru, polyalfabetický systém, Vigenérovská šifra, Kasiského metoda, Jeffersonův válec, nekonečný klíč.

Doba potřebná ke studiu: 5 hodin

Průvodce studiem

Studium této kapitoly je nutné k pochopení funkce transkripčních kryptografických systémů. Je prezentován historický průřez jejich vývojem a dokumentovány příklady různých kryptografických systémů včetně jejich slabých stránek, Na závěr je prezentován princip proudové šifry, která je použitelná i v dnešní době.
Na studium této části si vyhraďte pět hodin. Vyzkoušejte si, jak fungují různé transpoziční systémy i způsob jejich narušení.

transkripční systém

homofonie

Jeden z nejstarších ***transkripčních systémů*** pochází z Indie. Místo písmen otevřené zprávy se psala písmena, která podobně znějí. Např. v češtině by věta „*Včera se náš oddíl pokusil ustoupit.*“ mohla znít třeba: „*Fšera ze ňaf ottiv pokufyl huftoupyt.*“ Tyto systémy založené na ***homofonii*** (stejnoznělosti) hlásek již dnes nemají valné použití.

monoalfabetický systém

11.1 Monoalfabetické systémy

11.1.1 Systém Cézarovských šifer

Cézarovská šifra

Systém ***Cézarovských šifer*** je pozoruhodně jednoduchý (z našeho pohledu) a vydržel pozoruhodně dlouho. Používal ho Gaius Julius Caesar tak, že místo znaků původní zprávy psal znak o tři místa v abecedě dále. Jeho pokračovatel Marcus Antonius ji pro sebe upravil tak, že psal znak následující. V obou případech nebyla šifra nikdy prozrazena.

kryptografická transformace

Tuto kryptografickou transformaci můžeme popsat vztahem:

$$C_k: Z_N \to Z_N, \quad C_k(n) \equiv (n + k) \bmod N, \tag{81}$$

kde je
- n – znak původní zprávy,
- $C_k(n)$ – znak šifrované zprávy,
- k – klíč šifry, posunutí v abecedě,
- N – počet znaků abecedy.

Pro její dešifrování slouží transformace:

$$C_k{}^{-1}: Z_N \to Z_N, C_k(n)^{-1} \equiv (n + N - k) \bmod N, \tag{82}$$

Její vyluštění je nejjednodušší pomocí znalosti statistické pravděpodobnosti výskytu jednotlivých znaků českého jazyka v běžných textech, jak je uvádí tab. 1 a 2. Použijeme následující postup:

1. Určíme pravděpodobnosti výskytu znaků ve zprávě, jako podíly počtu jejich výskytů ku počtu všech znaků zprávy.
2. Sestavíme histogram rozložení těchto pravděpodobností ve stejném měřítku, jako histogram průměrného českého textu dané abecedy a porovnáme ho s histogramem zprávy.
3. Najdeme posunutí s největší shodou otevřené řeči se zprávou, a tím určíme šifrovací klíč.

Příklad 11.1: Máme za úkol dešifrovat následující text, o kterém předpokládáme, že byl šifrován pomocí nějaké Cézarovské šifry v abecedě Z_{26}:

Y O L X U B G T O S J A I N G Y B K N U V U Y O R
A P K S K I O Q N R A H O T G S F G Z X G I K T O
Y X G F O S K Z K J

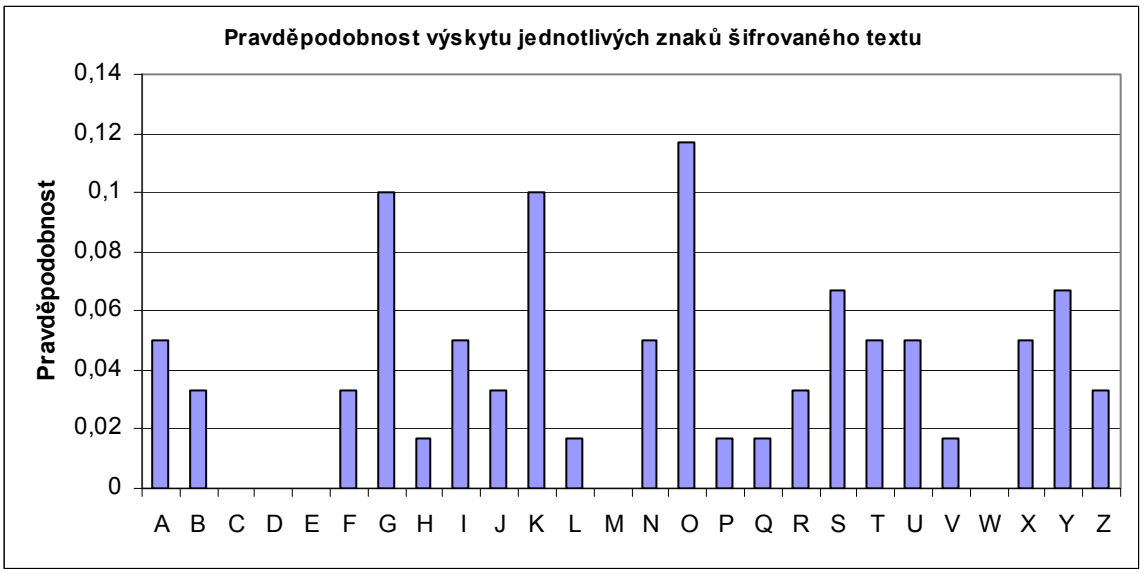
Nejprve tedy určíme pravděpodobnosti výskytů jednotlivých znaků v šifrované zprávě, jako podíl počtu výskytů těchto znaků ku celkovému počtu znaků ve zprávě.

Tab. 5. Pravděpodobnosti výskytu jednotlivých znaků v šifrované zprávě

A	**0,0500**	**J**	**0,0333**	**S**	**0,0667**
B	**0,0333**	**K**	**0,1000**	**T**	**0,0500**
C	**0,0000**	**L**	**0,0167**	**U**	**0,0500**
D	**0,0000**	**M**	**0,0000**	**V**	**0,0167**
E	**0,0000**	**N**	**0,0500**	**W**	**0,0000**
F	**0,0333**	**O**	**0,1167**	**X**	**0,0500**
G	**0,1000**	**P**	**0,0167**	**Y**	**0,0667**
H	**0,0167**	**Q**	**0,0167**	**Z**	**0,0333**
I	**0,0500**	**R**	**0,0333**		

Z hodnot tab. 5 sestavíme histogram výskytu znaků v šifrované zprávě.

Tab. 6. Histogram šifrovaného textu



Jelikož kryptografická transformace spočívá v posouvání znaků v abecedě, můžeme se pokusit najít klíč šifry posouváním histogramů šifrovaného textu v tab. 6 po histogramu českých textů v tab. 2. Text je však velmi krátký a blízkost histogramů nemusí být průkazná. Pokusme se tedy vyjádřit míru shody při různých posunutích matematicky, jako součet kvadrátů odchylek mezi pravděpodobností z tab. 5 šifrovaného textu, ale po posunutí o dešifrovací klíč a pravděpodobností z tab. 1 českého textu:

$$I_s(k) = \sum_{i=0}^{N-1} \left[P_o(y_i) - P_s\left[(y_i + k) \bmod N \right] \right]^2 , \qquad (83)$$

kde je $I_s(k)$ – index odchylky pro klíč k,
P_o – pravděpodobnost výskytu znaku českého textu,
P_s – pravděpodobnost výskytu znaku šifrovaného textu,
y_i – znaky použité abecedy, $i = 0, 1, \ldots N - 1$.

Tento index odchylky bude nejmenší tam, kde je shoda v posunutí histogramu šifrované zprávy nejbližší histogramu českého textu.

Index odchylky je však závislý také na délce dešifrovaného textu. Tab. 7 udává jeho hodnoty pro všechny dešifrovací klíče a několik délek dešifrované zprávy. Pro správné dešifrování bylo nutné použít alespoň 20 znaků dešifrované zprávy. Pro kratší zprávy nebylo minimum indexu odchylky určujícím.

Tab 7. Tabulka indexu odchylky pro různé délky dešifrované zprávy

klíč	10	15	20	25	60
0	0.1150	0.0738	0.0725	0.0712	0.0573
1	0.1087	0.0755	0.0682	0.0669	0.0531
2	0.1128	0.0713	0.0648	0.0684	0.0553
3	0.1016	0.0734	0.0671	0.0732	0.0534
4	0.1462	0.0883	0.0892	0.0880	0.0630
5	0.1286	0.0794	0.0721	0.0676	0.0469
6	0.0836	**0.0527**	0.0426	0.0388	0.0482
7	0.0810	0.0558	0.0595	0.0680	0.0506
8	0.1312	0.0890	0.0837	0.0856	0.0625
9	0.1345	0.0916	0.0855	0.0729	0.0608
10	0.1072	0.0756	0.0607	0.0584	0.0550
11	0.1022	0.0696	0.0742	0.0788	0.0526
12	0.0852	0.0530	0.0592	0.0600	0.0398
13	0.1362	0.0876	0.0782	0.0688	0.0620
14	0.1155	0.0869	0.0820	0.0805	0.0671
15	0.1041	0.0792	0.0766	0.0749	0.0575
16	**0.0676**	0.0587	0.0501	0.0424	0.0309
17	0.1175	0.0663	0.0515	0.0533	0.0633
18	0.1325	0.0929	0.0905	0.0901	0.0749
19	0.0996	0.0751	0.0711	0.0670	0.0582
20	0.0861	0.0532	**0.0401**	**0.0365**	**0.0134**
21	0.1095	0.0609	0.0620	0.0629	0.0476
22	0.1116	0.0727	0.0791	0.0760	0.0557
23	0.1126	0.0894	0.0866	0.0772	0.0630
24	0.1007	0.0648	0.0607	0.0585	0.0436
25	0.1058	0.0759	0.0693	0.0696	0.05698

Uvedený postup je možno velmi snadno algoritmizovat a zpracovat na počítači.

11.1.2 Systém afinních šifer

afinní šifra

Systém afinních šifer tvoří transformace:

$$A_{a,k}: Z_N \to Z_N,\ A_{a,k}(n) \equiv (a.n + k) \bmod N, \qquad (84)$$

kde je a, k – klíč šifry, a i k jsou přirozená čísla, a je nesoudělné s N.

Klíč afinní šifry tedy tvoří dvojice $[a, k]$, přitom jsou zvláštními případy šifry:
$[a, 0]$ – nazývá se ***multiplikativní šifra***,
$[1, k]$ – afinní šifra degeneruje na Cézarovskou šifru.

multiplikativní šifra

Řešení afinní šifry spočívá v nalezení inverzní transformace ve tvaru:

$$A_{\tilde{a},-\tilde{a}.k} : Z_N \Rightarrow Z_N, A_{\tilde{a},-\tilde{a}.k}(n) \equiv (\tilde{a}.n - \tilde{a}.k + N) \bmod N, \qquad (85)$$

kde $\tilde{a}$ označuje jednoznačně určený prvek $\tilde{a} \in N$, pro který platí:

$$(a.\tilde{a}) \bmod N = 1. \qquad (86)$$

Protože prvky a klíče afinní šifry musí být nesoudělné s N, musí patřit do následujících množin:

$$Z_{26} = \{1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25\}, \qquad (87a)$$

$$Z_{27} = \{1, 2, 4, 5, 7, 8, 10, 11, 13, 14, 16, 17, 19, 20, 22, 23, 25, 26\}. \qquad (87b)$$

K nim inverzní prvky pak uvádějí následující tabulky:

a	$\tilde{a}$	a	$\tilde{a}$	a	$\tilde{a}$
1	1	9	3	19	11
3	9	11	19	21	5
5	21	15	7	23	17
7	15	17	23	25	25

a	$\tilde{a}$	a	$\tilde{a}$	a	$\tilde{a}$
1	1	10	19	19	10
2	14	11	5	20	23
4	7	13	25	22	16
5	11	14	2	23	20
7	4	16	22	25	13
8	17	17	8	26	26

Postup řešení si ukažme na následujícím příkladu.

Příklad 11.2: Máme za úkol dešifrovat následující text, o kterém předpokládáme, že byl šifrován pomocí nějaké afinní šifry v abecedě Z_{26}:

G H A N K U F Z E F N I X H Q N A N A H K U B E H
Q B G H Z E Q B S J C H G H W F Z B M U N U F I Q
J Z E H A N

Nejprve tedy určíme počty výskytů jednotlivých znaků v šifrované zprávě. Stejně tak bychom mohli určit pravděpodobnosti výskytů jednotlivých znaků v šifrované zprávě.

Tab. 8. Výskyt jednotlivých znaků v šifrované zprávě

A	**4**	**J**	**2**	**S**	**1**
B	**4**	**K**	**2**	**T**	**0**
C	**1**	**L**	**0**	**U**	**4**
D	**0**	**M**	**1**	**V**	**0**
E	**4**	**N**	**6**	**W**	**1**
F	**4**	**O**	**0**	**X**	**1**
G	**3**	**P**	**0**	**Y**	**0**
H	**8**	**Q**	**4**	**Z**	**4**
I	**2**	**R**	**0**		

Znaky šifrované zprávy nyní seřadíme podle počtu výskytů počínaje nejčastějším a srovnáme s pořadím četnosti znaků v otevřené řeči:

8	6	4	3	2	1	0
H	N	B Q A F U Z E	G	K J I	S X C M W	R L V P O T Y D
E	A	I N O S T P U	D	K L M	C Z J R V	H YB FGQWX

Odtud předpokládáme, že při šifrování byly použity vztahy:
$E \to H, \quad A \to N$

Tento předpoklad bychom mohli ověřit porovnáním nejčastěji se vyskytujících dvojic znaků v textu apod.

kongruence

Nyní řešíme soustavu ***kongruencí***, kde ***kongruence modulo n***, kde n je přirozené číslo, je relace $\equiv$ na množině celých čísel $\mathbf{Z}$ splňující vztah: $x \equiv y$ právě tehdy, když existuje $k \in \mathbf{Z}$ tak, že $kn = (x - y)$ (tj. n dělí $(x - y)$).

$$\begin{aligned} 7 &\equiv a.4 + k \bmod 26 \qquad (E \to H), \\ 13 &\equiv a.0 + k \bmod 26 \qquad (A \to N). \end{aligned} \tag{88}$$

K řešení kongruencí je potřeba znalostí, které přesahují rozsah této publikace. Zájemci naleznou dostatek podkladů v literatuře. V tomto příkladě jsme schopni najít řešení i zkusmo. Z druhé kongruence je jasná hodnota $k = 13$. Pro tuto hodnotu nyní zkusíme dosazovat do první kongruence všechny možné hodnoty parametru a podle (87a), z nich vyhovuje jediná hodnota $a = 5$. Nalezli jsme šifrovací klíč afinní šifry $a = 5$, $k = 13$, k němu přísluší klíč dešifrovací: $\tilde{a} = 21$, $k = 13$. S jeho pomocí text dešifrujeme.

Poznámka: Jak již bylo zmíněno, řešení kongruencí není běžně známý aparát. Tento příklad nechť poslouží zájemcům pro hlubší zvládnutí problematiky kryptologie. Ti mohou stejný postup uplatnit také pro řešení Cézarovských šifer. Pro méně matematicky zanícené je možno použít postup pro řešení obecné monoalfabetické šifry, jak bude uveden dále. Ukázkový text tohoto příkladu je však velice krátký a řešení tímto postupem pro tak krátké texty nemusí vést ke správnému cíli.

11.1.3 Obecná monoalfabetická šifra

obecná monoalfabetická šifra

Můžeme snadno odvodit, že existuje celkem $N!$ různých přiřazení abecedy otevřeného textu abecedě šifrovaného textu. Např.:

$26! = 403\,291\,461\,126\,605\,635\,584\,000\,000.$

klíčová fráze
heslo

heslem míchaná abeceda

Mohli bychom tedy náhodně vygenerovat jedno z přiřazení a použít pro šifrování. Klíčem je však celé přiřazení, tedy $2 \cdot N$ znaků, které musíme předat tajně. To je samozřejmě nevýhodné. Proto obvykle hledáme způsoby, jak vystačit s kratším klíčem. Jedním ze způsobů je použití ***klíčové fráze*** (hesla). Tak vzniká tzv. ***heslem míchaná abeceda***. Jejím základem je klíč – heslo absahující alespoň 10 vzájemně různých znaků.

Zvolme např. heslo DOBRY CLOVEK pro abecedu Z_{26}.

V něm vynecháme všechny opakující se znaky. Prvních deset znaků pak zapíšeme do řádku a doplníme je ostatními nepoužitými znaky abecedy do schématu:

D	O	B	R	Y	C	L	V	E	K
A	F	G	H	I	J	M	N	P	Q
S	T	U	W	X	Z				

Protože ve druhém a třetím řádku jsou si znaky velmi blízké, budeme nyní brát znaky po sloupcích a přiřazovat jednotlivým znakům původní zprávy:

A	B	C	D	E	F	G	....	znaky původní zprávy
↓	↓	↓	↓	↓	↓	↓	...	
D	A	S	O	F	T	B	...	znaky šifrované zprávy

Použitím klíče o délce 11 znaků jsme získali jedinou monoalfabetickou šifru. Pro dešifrování postačí předat toto, poměrně krátké heslo.

Úloha narušitele při luštění šifry bez znalosti klíče bude již dosti komplikovaná. Kromě exaktních matematických metod používáme řadu intuitivních postupů, zejména:

- Pokud byly zachovány mezery mezi slovy nebo jsme schopni je jednoznačně rozpoznat (mají výrazně častější výskyt), začneme analyzovat kratší slova, která poskytují menší prostor pro kombinace.
- Hledáme charakteristické kombinace znaků (dvojice, trojice). Nejčastěji se vyskytují na začátcích a koncích slov.

- Pokusíme se odhadnout charakteristická slova, která se mohou v textu vyskytovat. (Podle oboru zprávy, autora apod.)
- Pokusíme se odhalit a separovat samohlásky a souhlásky neboť:
  - samohlásky mají tendenci se obklopovat souhláskami,
  - znaky, které mají malý počet různých sousedů jsou obvykle souhlásky (výjimkou je ch a ou), tito sousedi jsou obvykle samohlásky,
  - jestliže nějaká dvojice vystupuje také často v opačném pořadí, pak jeden z těchto znaků je obvykle samohláska,
  - prakticky v každém „běžném“ slově vystupuje samohláska (nebo r, l).

Při luštění můžeme využít následující informace [79]:

	nejčastější česká slova	nejčastější dvojice hlásek	nejčastější začátky slov
1	a	je	te
2	být	st	po
3	ten	ne	ne
4	v(ve)	na	pro
5	on	po	a
6	na	se	ob
7	že	ní	ta
8	s(se)	ro	ja
9	z(ze)	en	vi
10	který	le, em, la, ov, li, to, ko, te, el, pr	u

Příklad 11.3: Máme za úkol dešifrovat následující text, u kterého jsme zjistili, že nebyla použita Cézarovská, ani afinní šifra. (Znovu upozorňujeme, že text je velmi krátký, za normálních okolností by zřejmě nebyl řešitelný.)

XIQRFQDQXIDQRFQMFKQDQRFIQXIUQRLX_Q
DQAPMQLMDWFQVQRFOUIFYQXHDYKUH_QLMDMQYXGX_Q

Výskyt znaku mezera (byl zastoupen znakem **_**) naznačuje použití abecedy Z_{27}.

Určeme tedy výskyty znaků:

Tab. 9. Výskyt jednotlivých znaků v šifrované zprávě

A	**1**	**J**	**0**	**S**	**0**
B	**0**	**K**	**2**	**T**	**0**
C	**0**	**L**	**3**	**U**	**3**
D	**7**	**M**	**5**	**V**	**1**
E	**0**	**N**	**0**	**W**	**1**
F	**7**	**O**	**1**	**X**	**7**
G	**1**	**P**	**1**	**Y**	**3**
H	**2**	**Q**	**18**	**Z**	**0**
I	**5**	**R**	**5**	**–**	**3**

Srovnejme nejčastěji se vyskytující znaky

18	7	5	3	2	1	0
Q	D F X	I M R	L U Y _	H K	A G O P V W	B C E J N S T Z
–	E A	I N O S	T P U D	K L M	C Z J R V	H Y B F G Q H X

Výskyt znaku Q je natolik častý, že zjevně přísluší původnímu znaku mezera. Přepíšeme tedy zprávu rozdělenou do slov a najdeme nejčastěji se vyskytující slova:

XI RF D XID RF MFK D RFI XIU RLX_
D APM LMDWF V RFOUIFY XHDYKUH_ LMDM YXGX_

Nejčastější výskyt slov

jednopísmenná	dvoupísmenná	třípísmenná
D –3 x	RF – 2x	XID – 1x
V – 1x	XI – 1x	MFK – 1x
		RFI – 1x
		XIU – 1x
		APM – 1x

Nejčastější dvojice znaků: RF, LM, XI

Odhalené transformační vztahy si budeme ihned zapisovat do tabulky:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	_
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
																										Q

Současně sestavíme tabulku pro heslem míchanou abecedu i když nemáme žádné informace o jejím použití. Všechny poznatky si budeme ihned zapisovat do všech tabulek i do textu.

A	D	G	J	M	P	S	V	X	Z
B	E	H	K	N	Q	T	W	Y	- **Q**
C	F	I	L	O	R	U			

Vynecháme-li všechny chybné úvahy, dospějeme k cíli např. následujícím postupem:

1. Podle četnosti výskytu očekáváme, že E, A → D, F, X. Přitom znak D se vyskytuje jako samostatné slovo, tedy platí: A → D.
2. Znak F se vyskytuje často jako druhý v pořadí, je tedy zřejmě samohláskou a platí: E → F.
3. Další jednopísmenné slovo je V, tedy platí V → V.
4. Častá je dvojice JE → RF, jako samostatné slovo i v začátcích slov, tedy: J → R.
5. Častá dvojice znaků XI nemůže být slovem VE, spíše tedy druhým nejčastějším slovem ON: O → X, N → I.

6. Slovo XIU začíná na ON, je to tedy osobní zájméno, nejspíše ONI: I → U.
7. Častá dvojice znaků je LM, nejspíš tedy ST → LM: S → L, T → M.
8. Z významu textu zřejmě platí TEZ → MFK: Z → K,
   JSOU → RLX_: U → _.
9. Slovo APM končí na T, zřejmě je to slovo BYT: B → A, Y → T.
10. Z významu textu zřejmě platí STALE → LMDWF: L → W,
    JEDINEM → RFOUIFY: D → O, M → Y,
    OKAMZIKU → XHDYKUH_: K → H,
    MOHOU → YXGX_: H → G.

Tím je celý text dešifrován a vše nasvědčuje tomu, že byla skutečně použita heslem míchaná abeceda. V tabulce heslem míchané abecedy však chybí znaky, které nebyly ve zprávě použity:

A **D**	D **O**	G	J **R**	M **Y**	P	S **L**	V **V**	X	Z **K**
B **A**	E **F**	H **G**	K **H**	N **I**	Q	T **M**	W	Y **P**	- **Q**
C	F	I **U**	L **W**	O **X**	R	U **_**			

Pokusme se celou tabulku doplnit podle pravidel pro její tvorbu:

1. Mezi znaky A a F není volné místo, takže znaky B, C, E musí být součástí hesla.
2. Mezi znaky I a M je jedno místo, protože K i L je již v hesle, patří sem J.
3. Mezi znaky M a P je jedno místo, protože O je v hesle, patří sem N.
4. Mezi Q a U jsou dvě místa, protože R je v hesle, patří sem po řadě S a T.
5. Mezi znaky X a mezera je jedno místo, protože Y je v hesle, patří sem Z.
6. Heslo je třeba doplnit o znaky BCE, může tedy jít o heslo DOBRÝ ČLOVĚK. To však nelze nijak ověřit. Jedině zachycením další, stejně šifrované zprávy.

Rozvoj kryptografických systémů založených na monoalfabetických šifrách spadá do začátku 15. století, kdy vznikají jednoduché substituce zvané nomenklátory. Staly se základním prostředkem šifrování na téměř 450 let – až do vynálezu telegrafu.

nomenklátor

Nomenklátory šly ještě trochu dále než obecná monoalfabetická šifra. Nomenklátor je v podstatě tajný kód, který jednotlivým znakům (ale také skupinám znaků nebo celým slovům) přiřazuje jiné skupiny znaků. Přitom je možné, aby nejčastější znaky měly více možných vyjádření v šifrovaném textu. Tím se snažily odstranit statistické závislosti textu, i když tehdy ještě kryptologie nepoužívala tak propracovaný matematický aparát jako dnes.

11.2 Autoklíč

autoklíč
autokláv

Princip ***autoklíče*** (***autoklávu***) popsal Francouz Blaise de Vigenére. V té době již bylo známo, že monoalfabetická šifra je velmi nepříjemná skutečností, že zachovává statistické vlastnosti otevřeného textu. Proto vytvořil šifrovací systém, ve kterém znak šifrované zprávy závisel nejen na znaku původní zprávy, ale také na všech předchozích. Popsal ho ve své knize Traicté des Chiffres (Pojednání o šifrách) z r. 1585.

klíč

Postup byl jednoduchý. Nejprve zvolíme krátké heslo – ***klíč***. To přičteme postupně k jednotlivým znakům zprávy. Tím získáme první znaky šifrované zprávy. V dalším postupu se již práce liší. Buď použijeme k šifrování dalšího úseku zprávy začátek šifrované zprávy nebo začátek otevřené zprávy. Například heslo „Tajná zpráva“ máme zašifrovat pomocí autoklíče s klíčem „C“

Princip autoklíče z šifrované zprávy:

otevřená zpráva	T	A	J	N	A	Z	P	R	A	V	A
klíč	C	V	V	E	R	R	Q	F	W	W	R
šifrovaná zpráva	V	V	E	R	R	Q	F	W	W	R	R

Princip autoklíče z otevřené zprávy:

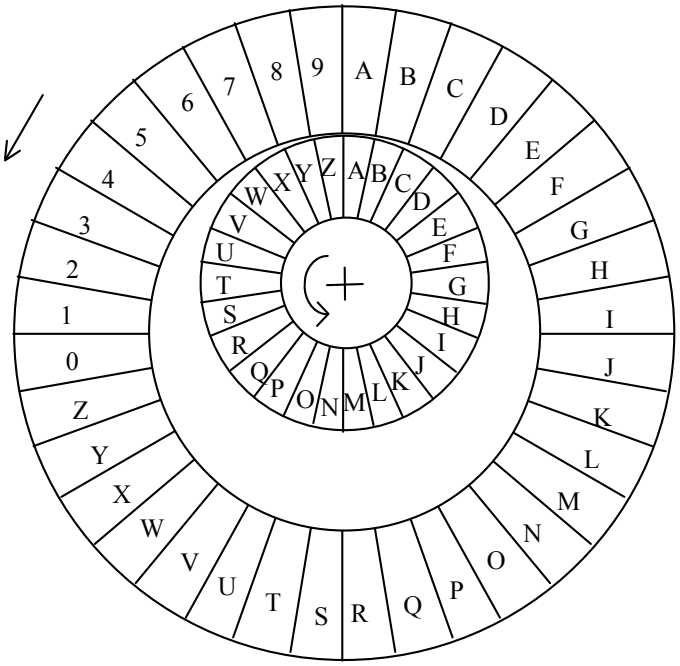
otevřená zpráva	T	A	J	N	A	Z	P	R	A	V	A
klíč	C	T	A	J	N	A	Z	P	R	A	V
šifrovaná zpráva	V	T	J	W	N	Z	O	G	R	V	V

Na svoji dobu byl tento systém velmi dobrý, má však v sobě skryto velké nebezpečí. Snadno si ho uvědomíme, pokud se budeme snažit dešifrovat zprávu, ve které se budou vyskytovat chyby.

kotouče různého průměru

Na podobném principu založil svoji šifru D. Wadsworth, když v r. 1817 sestavil mechanickou pomůcku. Sestávala ze ***dvou kotoučů různého průměru***. Vnitřní byl otočný s 26 znaky, vnější k němu přiléhal pomocí ozubení a měl 33 znaky. Na počátku se nastavil klíč, např. A $\rightarrow$ C znamená, že proti vnějšímu A se nastavilo vnitřní C, pak se otáčelo vnitřním kotoučem, až se v okénku objevil požadovaný znak. Do šifrované zprávy se opsal znak proti němu na vnějším kotouči. Pokud měl být vyslán dvakrát stejný znak, musel se vnitřní kotouč otočit o celou otáčku. Postup dešifrování je stejný, jen se nastavují znaky vnějšího kotouče a čtou na vnitřním.

Na obr. 14 je naznačen takový kotouč s 26 znaky vnitřního a 36 znaky vnějšího kotouče. Protože se nedostává písmen, jsou přidány číslice. Počáteční nastavení je provedeno A $\rightarrow$ A.



Obr. 14. Princip použití nestejných kotoučů

Tento systém vypadá jako velmi dobrý. Existuje celkem 26 x 36 = 936 různých počátečních nastavení. Ale bohužel jen 36 jich dává různé výsledné zprávy. Stačí si uvědomit, že pro první znak zprávy třeba F dá nastavení B → V stejný výsledek jako A → U. Je tedy jen tolik různých nastavení, proti kolika vnějším znakům můžeme nastavit vnitřní znak A.

Vyzkoušet všech 36 nastavení je jen otázkou času. Zkuste si to na zprávě:
H2GMW 8N8MS UYLOV 4C1ET APAOU 4GVGU 01IR0C KTH2Q R8P27 XE.

11.3 Polyalfabetické systémy

polyalfabetický systém

Všechny monoalfabetické šifry zachovávají původní četnosti výskytu znaků, a tím napomáhají snadnějšímu rozluštění. Tento nedostatek se snaží odstranit polyalfabetické šifry, které tvoří konečná (nebo nekonečná) posloupnost monoalfabetických šifer. Současně se však snažíme, abychom nemuseli předávat příliš mnoho informace nutné pro dešifrování textu. Proto vznikaly různé pomůcky pro šifrování, vedoucí až k šifrovacím strojům, jako známá ENIGMA za II. světové války.

11.3.1 Vigenérovské šifry

Vigenérovská šifra

Principem ***Vigenérovského systému*** je použití množiny Cézarovských šifer. Množina použitých Cézarovských šifer se přitom zadává obvykle jako textové heslo. Pozice jednotlivých znaků v hesle udávají parametry k pro jednotlivé Cézarovské šifry. Tyto šifry se pak cyklicky používají vždy pro zašifrování jednoho znaku zprávy. Systém nese paradoxně jméno člověka, který jej neobjevil, ale jen popsal ve své knize (zmíněné v předchozí kapitole). Vynalezl ho italský šlechtic Giovanni Battista Belaso a popsal v knize La cifra v roce 1553.

Pro klíč $k = \{k_1, k_2, ..., k_m\}$,

kde je k_i – klíč Cézarovské šifry, $k_i \in Z_N$, $i = 1, 2, ..., m$,
m – délka hesla,

je polyalfabetická transformace s množinou transformací

$$y_1 \equiv x_1 + k_1 \bmod N,$$

$$y_2 \equiv x_2 + k_2 \bmod N,$$

.....

$$y_m \equiv x_m + k_m \bmod N,$$

$$y_{m+1} \equiv x_{m+1} + k_{m+1} \bmod N,$$

Postup šifrování je následující:

Např pomocí hesla: **AHOJ** šifrujeme text: **NEMOHU PRIJIT** následovně:

N + A mod 26 = N	P + O mod 26 = D
E + H mod 26 = L	R + J mod 26 = A
M + O mod 26 = A	I + A mod 26 = I
O + J mod 26 = X	J + H mod 26 = Q
H + A mod 26 = H	I + O mod 26 = W
U + H mod 26 = B	T + J mod 26 = C

A dostáváme šifrovanou zprávu: **NLAXHB DAIQWC.**

Vigenérův čtverec

Pro zjednodušení práce slouží ***Vigenérův čtverec***, viz tab. 10. Písmeno otevřené zprávy najdeme na horním řádku, písmeno klíče na levém sloupci, v jejich průsečíku leží písmeno šifrované.

Tab. 10. Vigenérův čtverec

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Vigenérovskou šifru je samozřejmě možno dále komplikovat. Např. promícháním záhlaví nebo řádků Vigenérova čtverce pomocí heslem míchané abecedy apod.

Prvním krokem rozluštění je pak určení délky klíče, tedy periody jeho opakování. Na délku klíče může ukazovat také hodnota indexu koincidence $I(C)$, viz (80). Odtud by délka klíče měla odpovídat:

$$m \cong \frac{\left(\sum_{i=0}^{N-1} P^2(x_i) - \frac{1}{N}\right) \cdot n}{I(C) \cdot (n-1) - \frac{1}{N} \cdot n + \sum_{i=0}^{N-1} P^2(x_i)}. \tag{89}$$

Kasiského metoda

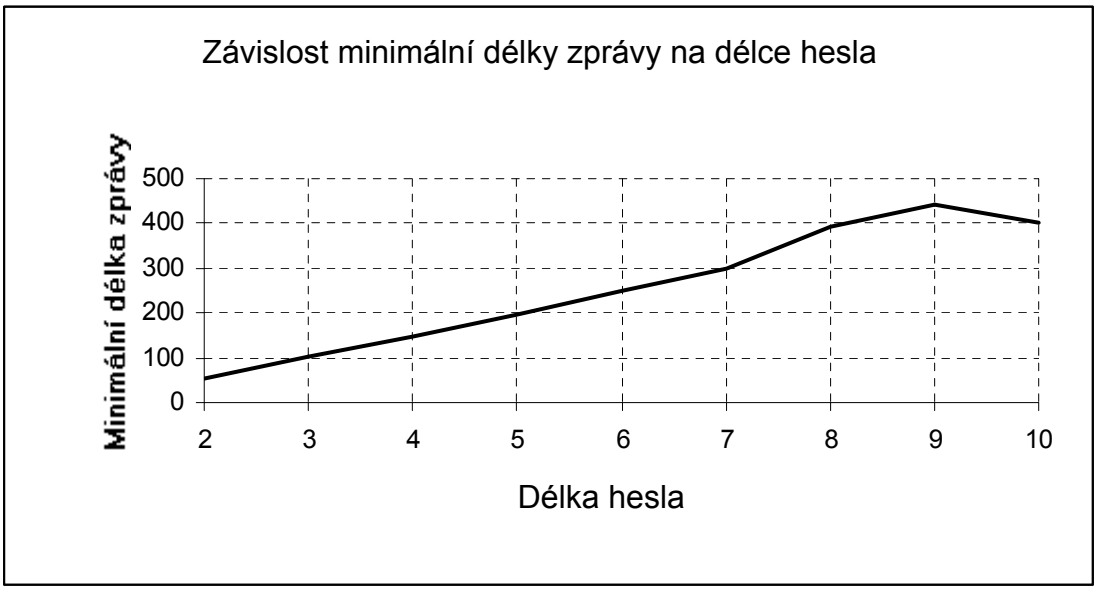
Odhad podle (89) je však jen velmi přibližný. K přesnějšímu zjištění délky klíče se používají jiné postupy, jako **Kasiského metoda**.

Tato metoda vychází z frekvence výskytu jednotlivých znaků a především jejich ucelených skupin v jazyce. Pokud se v textu vyskytují stejné skupiny znaků, pak se transformují na stejnou skupinu znaků šifry tehdy, jsou-li vzdáleny o celistvý násobek délky klíče. Hledáme tedy v šifrovaném textu stejné skupiny znaků a zjišťujeme jejich vzdálenost (o jedničku větší počet znaků mezi jejich začátky).

Je samozřejmé, že ne každá stejná skupina zprávy bude transformována do stejné skupiny znaků v šifře a naopak, ne každá stejná skupina znaků v šifře musí být skutečně stejnou skupinou i v původní zprávě. Čím je však stejná skupina delší, tím větší je pravděpodobnost, že je stejná i v původní zprávě.

Zjištěné skupiny zapíšeme do tabulky podle jejich délky a zjistíme společného dělitele co největšího jejich počtu. Tento společný dělitel pak bude nejspíše délkou použitého klíče.

Pokud se chceme vyslovit ke kvalitě Vigenérovského systému, musíme si uvědomit, že se jedná o vícenásobné použití Cézarovského systému, minimální délka zašifrovaného textu je tedy rovna zhruba dvacetinásobku délky hesla. Odhalení délky hesla však vyžaduje také nějakou délku šifrovaného textu. Rozborem většího počtu šifrovaných textů různých autorů byla za pomoci studentů zjištěna následující závislost minimální délky zprávy potřebné k rozluštění na délce hesla:



Obr. 15. Závislost minimální délky zprávy na délce hesla

Příklad 11.4: Máme řešit zašifrovanou zprávu následujícího znění. Index koincidence ukázal na polyalfabetickou šifru.

J O T T A	**N B R U U**	**U C U M M**	**Z F R Z E**	**C S W Q S**
S T N Z J	**O I A K E**	**X W M M L**	**U M T T I**	**M S Y Z I**
D C V V E	**X W W C T**	**X S J J Y**	**N S U S A**	**U Z R K E**
L M U I D	**O Z R B E**	**V S V L E**	**V Y H H P**	**B O E G Z**
K H X H P	**B O E I M**	**E G R J Y**	**D R X A T**	**S R U W U**
R O J J Y	**.....**			

Určíme opakující se skupiny znaků a jejich vzdálenosti:

Skupina	Vzdálenost
HPBOE	10
JJY	65
EXW	20
JO	29
TT	40
RU	114
UM	28

Skupina	Vzdálenost
KE	40
IM	64
ID	29
JY	15
ZR	10
EV	5

Nyní určíme společné dělitele co největšího počtu z nich, jsou to od nejčastějšího:

2 pro 10 skupin,
5 pro 8 skupin,
10 pro 7 skupin.

Jakmile máme určenu délku klíče, přistoupíme k dešifrování textu. Text se vlastně skládá z několika částí (jejich počet je roven délce použitého klíče), které jsou zašifrovány Cézarovskou šifrou. Rozdělíme tedy text na tyto části, dešifrujme je jako Cézarovské šifry a složíme zpět.

Pokud byl klíč délky 2 znaky, rozdělíme text na dvě skupiny:
1. skupina písmen v pořadí lichých:

JTABUUUMFZCWSTZOAEWMUTISZDVEWCXJYSSURE
MIORESLVHPOGKXPOIERYRASUUOJ

2. skupina písmen v pořadí sudých:

OTNRUCMZRESQSNJIKXMLMTMYICVXWTSJNUAZKL
UDZBVVEYHBEZHHBEMGJDXTRWRJY

Na jejich dešifrování použijeme program pro dešifrování Cézarovských šifer, který určí klíče a původní texty zprávy, jak bylo popsáno u řešení Cézarovských šifer:

1. skupina: klíč k = 0, text zní:

JTABUUUMFZCWSTZOAEWMUTISZDVEWCXJYSSUREMI
ORESLVHPOGKXPOIERYRASUUOJ

2. skupina: klíč k = 9, text zní:
FKEILTDQIVJHJEAZBODCDKDPZTMONKJAELRQBCLUQS
MMVPYSVQYYSVDXAUOKINIKP

Složením textů dostáváme výslednou zprávu:
JFTKAEBIULUTUDMQFIZVCJWHSJTEZAOZ....,
která je evidentně nesmyslná, klíč tedy neměl délku 2 znaky, vyzkoušíme klíč délky 5 znaků:

Text rozdělíme na pět skupin, které dešifrujeme:
1. skupina
JNUZCSOXUMDXXNULOVVBKBEDSR
klíč k = 10, text zní:
ZDKPSIENKCTNNDKBELLRARUTIH

2. skupina
OBCFSTIWMSCWSSZMZSYOHOGRRO
klíč k = 14, text zní:
ANOREFUIYEOIEELYLEKATASDDA

3. skupina
TRURWNAMTYVWJURURVHEXERXUJ
klíč k = 9, text zní:
KILINERDKPMNALILIMYVOVIOLEA

4. skupina
TUMZQZKMTZVCJSKIBLHGHIJAWJ
klíč k = 8, text zní:
LMERIRCELRNUBKCATDZYZABSOL

5. skupina
AUMESJELIIETYAEDEEPZPMYTUY
klíč k = 0, text zní:
AUMESJELIIETYAEDEEPZPMYTUE

Jejich složením dostáváme zprávu:
ZAKLADNIM UKOLEM PRI RESENI SIFER JE URCENI DELKY KLICE PRITOM NENI NUTNE ABY DELKA KLICE BYLA DELITELEM DELKY ZPRAVY ZATO ZPRAVA MUSI BYT DOSTI DLOUHA ALE

Tím byla šifra rozluštěna. Pro luštění skutečných Vigenérovských šifer by však bylo potřeba mít k dispozici mnohem delší šifrovaný text, řádově alespoň třicetinásobek délky klíče. Pokud bychom provedli promíchání abecedy v záhlaví Vigenérova čtverce, nebo promíchání jejích řádků, tedy zašifrování pomocí soustavy obecných monoalfabetických šifer, bude postup práce stejný, ale dešifrování jednotlivých monoalfabetických šifer bude mnohem složitější.

Jeffersonův válec

Na podobném principu byl založen ***Jeffersonův válec***. Vynalezl ho americký ministr zahraničních věcí Thomas Jefferson (později se stal prezidentem) na konci 18. století. Bohužel byl laik a nepoznal dokonalost tohoto systému, proto se nepoužíval.

Na válec umístil 26 kotoučů, na kterých byly náhodně rozmístěny znaky abecedy, na každém jinak. Při šifrování se kotouče nastavily tak, aby v jednom řádku dávaly text otevřené zprávy, šifrovaná zpráva se přečetla z následujícího řádku nebo kteréhokoliv jiného. Takto vlastně použijeme k šifrování soustavu 26 obecných monoalfabetických šifer.

11.3.2 Polyalfabetická šifra s nekonečným klíčem

nekonečný klíč

Dosud popsané kryptografické systémy byly vynalezeny a také úspěšně rozluštěny ještě před začátkem 20. století. V jeho průběhu doznaly šifrovací systémy značného rozšíření. Především to bylo nasazením výpočetní techniky, která umožnila provádět rychle velké množství operací a vyhodnotit řadu statistických údajů o dešifrovaném textu. Jako zajímavost si nyní uveďme šifrovací systém používaný sovětskou rozvědkou za II. světové války, zejména na území Evropy. Tento systém lze charakterizovat jako polyalfabetický systém s nekonečným klíčem [56].

Klíčem byl text dohodnuté knihy. Obě strany ji musely mít ve stejném vydání. Všechny znaky se převáděly na číslice vysílané v pěticích. Jedna pětice určovala počátek klíče, slovo obsahující alespoň deset různých znaků. Tato pětice byla zařazena na dohodnuté místo do vysílané zprávy. Měla tvar:
1 2 0 8 9, kde je:
12 – strana v knize,
08 – řádek na stránce,
9 – pozice slova na řádku.

Řekněme, že je to slovo **Dokumentarfilme**. Použijeme ho jako klíč pro heslem míchanou abecedu doplněnou o tečku. Ta by sloužila pro doplnění zprávy na předepsaný počet znaků.

Krok 1:	D	O	K	U	M	E	N	T	A	R
	B	C	F	G	H	I	J	L	P	Q
	S	V	W	X	Y	Z	.			

V prvním řádku očíslujeme písmena v pořadí, jak jdou za sebou v abecedě:

Krok 2:	**2**	**7**	**4**	**0**	**5**	**3**	**6**	**9**	**1**	**8**
	D	O	K	U	M	E	N	T	A	R
	B	C	F	G	H	I	J	L	P	Q
	S	V	W	X	Y	Z	.			

Řádky označíme dohodnutými čísly, např. čísly z 3., 7. a 9. sloupce:

Krok 3:	**2**	**7**	**4**	**0**	**5**	**3**	**6**	**9**	**1**	**8**
4	D	O	K	U	M	E	N	T	A	R
6	B	C	F	G	H	I	J	L	P	Q
1	S	V	W	X	Y	Z	.			

Každému písmenu přiřadíme kód složený z čísla sloupce a čísla řádku:

Krok 4:	A	B	C	D	E	F	G	H	I	J
	14	**26**	**76**	**24**	**34**	**46**	**06**	**56**	**36**	**66**
	K	L	M	N	O	P	Q	R	S	T
	44	**96**	**54**	**64**	**74**	**16**	**86**	**84**	**21**	**94**
	U	V	W	X	Y	Z	.			
	04	**71**	**41**	**01**	**51**	**31**	**61**			

Máme odeslat například oznámení: „Die Leibstandarte Adolf Hitler ist in Warschau eingetroffen.“ („Tělesná standarta Adolfa Hitlera dorazila do Varšavy.“) Pro radiodepeši text zkrátíme a převedeme písmena na čísla podle kroku 4.

Krok 5:	H	I	T	L	E	R	S	T	A	N	D	A	R	T	E
	56	**36**	**94**	**96**	**34**	**84**	**21**	**94**	**14**	**64**	**24**	**14**	**84**	**94**	**34**
	I	N	W	A	R	S	C	H	A	U					
	36	**64**	**41**	**14**	**84**	**21**	**76**	**56**	**14**	**04**					

Získaná čísla seřadíme do pětimístných skupin.

Krok 6:	**56369**	**49634**	**84219**	**41464**	**24148**
	49434	**36644**	**11484**	**21765**	**61404**

Šifrovaný text po kroku 6 je stále monoalfabetická a proto snadno dešifrovatelná. Proto z kroku 4 vezmeme pod písmeny vždy první z dvojice čísel (číslo sloupce nad písmenem v kroku 3) a dostaneme pomocnou tabulku:

Krok 7:	A	B	C	D	E	F	G	H	I	J
	1	**2**	**7**	**2**	**3**	**4**	**0**	**5**	**3**	**6**
	K	L	M	N	O	P	Q	R	S	T
	4	**9**	**5**	**6**	**7**	**1**	**8**	**8**	**2**	**9**
	U	V	W	X	Y	Z	.			
	0	**7**	**4**	**0**	**5**	**3**	**6**			

Z klíčové knihy vezmeme text začínající klíčovým slovem a převedeme ho do číselné podoby pomocí tabulky z kroku 7.

Krok 8:

D	O	K	U	M	E	N	T	A	R	F	I	L	M	E	
2	**7**	**4**	**0**	**5**	**3**	**6**	**9**	**1**	**8**	**4**	**3**	**9**	**5**	**3**	
S	I	N	D	B	E	L	E	G	T	W	E	R	D	E	N
2	**3**	**6**	**2**	**2**	**3**	**9**	**3**	**0**	**9**	**4**	**3**	**8**	**2**	**3**	**6**
A	B	E	R	R	A	S	C	H	W	I	E	D	E	R	
1	**2**	**3**	**8**	**8**	**1**	**2**	**7**	**5**	**4**	**3**	**3**	**2**	**3**	**8**	
F	R	E	I												
4	**8**	**3**	**3**												

Nyní budeme sčítat hodnoty z tabulek v krocích 6 a 8 modulo 10. (např. 3 + 9 = 12, píšeme 2).

zpráva z kroku 6:	56369	49634	84219	41464	24148
zpráva z kroku 8:	27405	36918	43953	23622	39309
znění telegramu	**73764**	**75542**	**27162**	**64086**	**53447**

zpráva z kroku 6:	49434	36644	11484	21765	61404
zpráva z kroku 8:	43823	61238	81275	43323	84833
znění telegramu	**82257**	**97872**	**92659**	**64088**	**45237**

Do telegramu je navíc třeba vložit na dohodnuté místo skupinu pro určení začátku klíče **12089**. Postup dešifrování je zřejmý. Provedeme kroky 1, 2, 3, 4, dále 7, 8. Od telegramu odečteme hodnoty z tabulky získané v kroku 8 a převedeme čísla na písmena podle tabulky v kroku 4.

Kontrolní otázky:

39. Jaký je základní princip činnosti transkripčních kryptografických systémů?
40. Jaký je princip monoalfabetických systémů a jaká největší slabina těchto systémů?
41. Jaký je princip polyalfabetických systémů, jaké jsou možnosti určení délky použitého hesla?
42. Jsou nějaké polyalfabetické systémy použitelné i v dnešní době?

Shrnutí obsahu kapitoly

V této kapitole jsme se seznámili s typickými příklady transkripčních kryptografických systémů od historicky nejstarších Cézarovských systémů po nejkomplikovanější obecnou monoalfabetickou šifru. je ukázáno, že takové systémy nejsou použitelné v dnešní době. Dále jsou ukázány principy polyalfabetických systémů až po polyalfabetické systémy s nekonečným klíčem, používané za druhé světové války. Ty ukázaly cestu k dodnes používaným systémům.

12 Současné systémy s tajným klíčem

Cíl:

Cílem celé kapitoly je seznámení se současnými kryptografickými systémy. Je prezentován proudový šifrovací systém a vysvětleno proč je označován za nerozluštitelný. Následně jsou prezentovány dva šifrovací standardy DES a AES, které představují principy současných blokových šifrovacích systémů s tajným klíčem.

Po jejím prostudování byste měli být schopni:

- Popsat realizaci proudové šifry a vysvětlit proč je označována za nerozluštitelnou.
- Popsat funkci současných blokových šifrovacích systémů s tajným klíčem.

Klíčová slova této kapitoly:

Bezpečná šifra, proudová šifra, Vernamův systém, ROT13, systém DES, runda, vícerundový systém, Feistelův systém, systém RC4, systém RC5, výtah zprávy, hash, systém RC6, systém AES.

Doba potřebná ke studiu: 4 hodiny

Průvodce studiem

Studium této kapitoly je nutné k pochopení funkce současných kryptografických systémů. Je prezentován princip funkce proudové šifry a důvod proč je označována za nerozluštitelnou. Následně jsou představuje blokové šifrovací systémy DES a AES a jejich realizace.
Na studium této části si vyhraďte čtyři hodiny.

bezpečná šifra

S rozvojem výpočetní techniky poklesla bezpečnost klasických kryptografických systémů v zásadě na nulu. Díky možnosti velmi rychle vyhodnotit statistické vlastnosti textu je možno velmi úspěšně klasické systémy luštit, což bylo v předchozím textu uvedeno. Nutnost ochrany dat se však stále zvyšovala, proto vznikaly složitější kryptografické systémy. Mezi nimi byla objevena také ***bezpečná*** (nerozluštitelná) ***šifra*** [49].

proudová šifra

V roce 1917 zaměstnanec americké telegrafní společnosti A.T.&T. Vernam vynalezl svůj šifrovací systém, založený na bázi tzv. ***proudové šifry***. Při přenášení zpráv měl text na telegrafní pásce binární podobu a Vernam tuto skutečnost využil pro šifrování. K šifrování použil další, stejně dlouhou pásku, která obsahovala klíč, také v binární podobě. Z obou pásek bral postupně bit po bitu a sčítal je v binární podobě (ekvivalent použití operace XOR). Páska s heslem byla vyhotovena ve dvou provedeních, jednu použil odesílatel, druhá byla bezpečným kanálem předána příjemci. Každá páska se používala jen jednou a pak byla zničena. Jedná se vlastně o šifru s nekonečným klíčem, která je bez jeho znalosti skutečně nerozluštitelná.

Vernamův systém

Na základě ***Vernamovy proudové šifry*** jsou stavěny mnohé, především amatérské, kryptografické systémy. Nutnost generování klíčů nahrazují obvykle zabudováním generátoru náhodných čísel a tajné heslo používají pro jeho inicializaci. Tím je ovšem kvalita myšlenky značně omezena. Nedokonalost generátoru dává mnoho informace pro dešifrování systému, viz např. [15]. Totéž platí, pokud stejné heslo použijeme opakovaně. Ukažme si to na příkladu textu kódovaného mezinárodním telegrafním kódem CCITT-2, viz tab. 11. Jak vidíme, některá kódová slova nemají význam písmene, což nám usnadní luštění.

Tab. 11. Mezinárodní telegrafní abeceda CCITT-2

CCITT	znak	kód	CCITT	znak	kód
1	A _	11000	17	Q 1	11101
2	B ?	10011	18	R 4	01010
3	C :	01110	19	S !	10100
4	D vz	10010	20	T 5	00001
5	E 3	10000	21	U 7	11100
6	F	10110	22	V =	01111
7	G	01011	23	W 2	11001
8	H	00101	24	X /	10111
9	I 8	01100	25	Y 6	10101
10	J zv	11010	26	Z +	10001
11	K (	11110	27	návrat válce	00010
12	L)	01001	28	posun řádku	01000
13	M .	00111	29	číslicová změna	11011
14	N ,	00110	30	písmenová změna	11111
15	O 9	00011	31	mezera	00100
16	P 0	01101	32		00000

Příklad 12.1: Byly přijaty tři šifrované texty, pro které byl použit stejný klíč:

```
10111 10100 01100 11100 10111 00010 11001 10010
00110 11100 00111 01111 11011 00010 00001 10010
00100 10100 01100 00110 10001 00000 10011 11010
```

Tučně jsou vyznačeny shodné znaky. Při použití stejného klíče pro všechny zprávy musí nutně jít také o stejné znaky v otevřených zprávách. Ve skutečnosti bychom podle těchto shod usuzovali na použití stejného klíče pro všechny zprávy a ne naopak. Z dosavadních znalostí předpokládáme, že zprávy obsahují tři různá jména (křestní), nejspíše dívčí.

Takový předpoklad není nijak překvapivý. I v praxi bychom mohli předpokládat jakého tématu se šifrovaný text týká, z jaké oblasti pochází apod. Napomáhají tomu zejména často se opakující telegramy shodného obsahu (typické je hlášení „není co hlásit“) či shodné struktury (bankovní příkaz). Velké množství telegramů obsahuje v záhlaví označení adresáta a na konci podpis odesílatele a ti jsou často známí apod.

Postup řešení může být různý. Například zjistíme z kalendáře běžná dívčí jména na osm znaků:

jméno	**den**	**měsíc**	**jméno**	**den**	**měsíc**
ZDISLAVA	29	1	KRISTYNA	24	7
VERONIKA	7	2	OLDRISKA	6	8
GABRIELA	8	3	MICHAELA	19	10
VIKTORIE	10	3	STEPANKA	31	10
SVETLANA	20	3	MAHULENA	17	11
PATRICIE	2	7	KATERINA	25	11
KAROLINA	14	7	BOHUMILA	28	12

Nyní můžeme např. pro první zprávu zkoušet jednotlivé předpokládané texty a ověřovat, zda takto získané heslo dává smysluplnou zprávu i u ostatních textů. Nesmíme přitom zapomenout, že náš seznam předpokládaných textů nemusí být úplný!

Tímto postupem zjistíme, že jméno KATERINA dává použitý klíč:

```
01001 01100 01101 01100 11101 01110 11111 01010
```

a ten poskytuje u obou zbývajících šifrovaných zpráv smysluplné výsledky.
Druhou možností je zkoušet všechny možné klíče, jejich počet však s rostoucím počtem znaků prudce roste, takže na běžně dostupné technice lze v rozumném čase zvládnout vyzkoušení všech možných klíčů pro několik desítek znaků. Ihned odmítneme klíče, které v některé ze zpráv produkují znaky neodpovídající písmenům. V ostatních skupinách budeme vybírat ty klíče, které produkují zprávy nejvíce odpovídající našim předpokladům. Postup je obecnější, také ovšem časově náročnější, ale dospějeme ke stejnému výsledku.

ROT13

O tom, jak slabé systémy se v současnosti používají, svědčí „kódování ***ROT13***“, používané v počítačové síti Internet v diskusním systému USENET [13]. Jedná se o Cézarovský systém s klíčem $k = 13$, který jak známo prakticky žádnou ochranu neposkytuje. Použití ROT13 má ale jiný význam. Slouží jen k tomu, aby zpráva nebyla přímo čitelná, pokud by její obsah mohl být pro některé čtenáře pohoršující. Odtud plyne také označení systému jako ***kódování***, neboť kódová kniha je veřejně známa.

12.1 Systém DES

systém DES

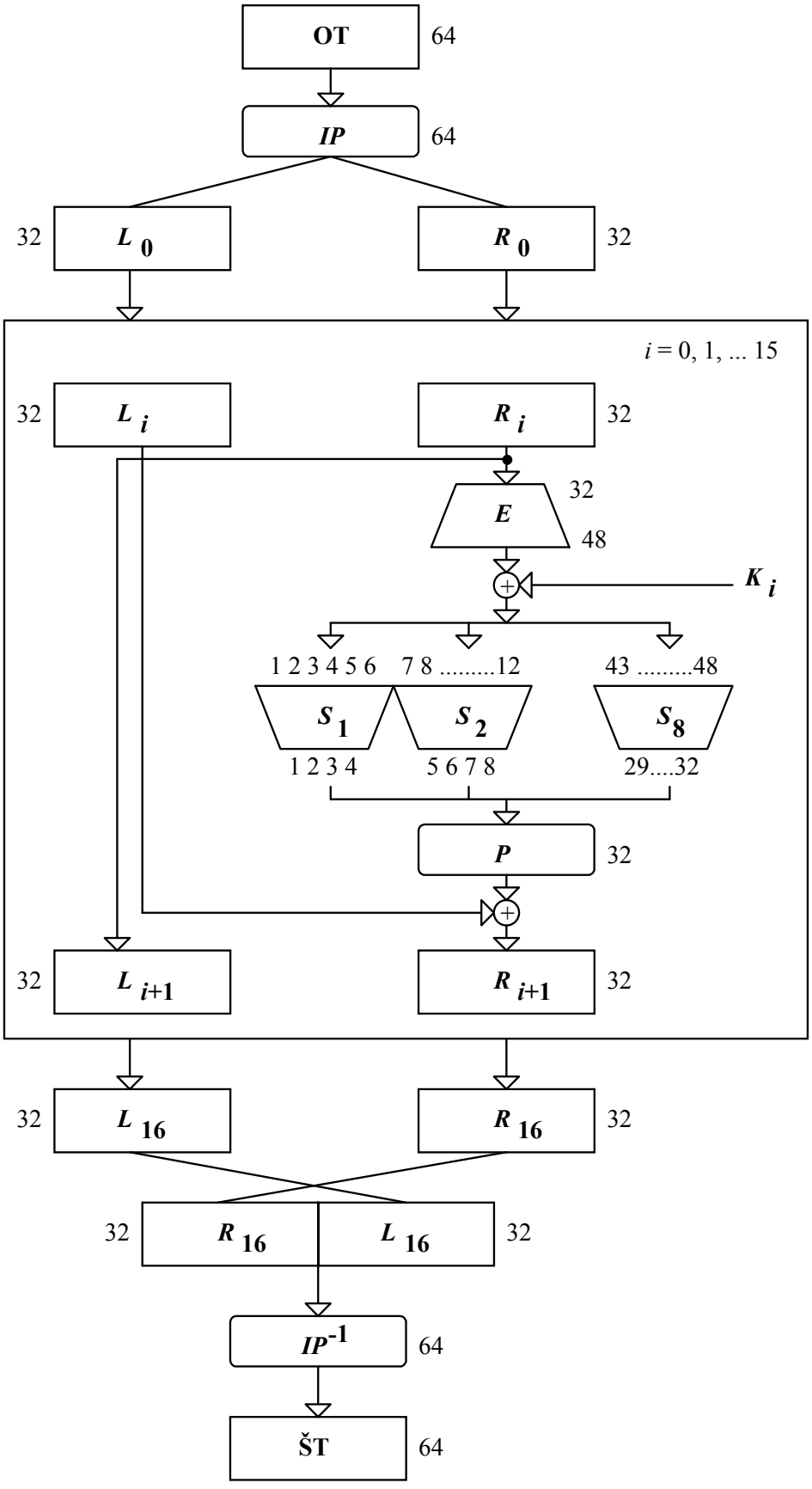
Šifrovací systém DES (Data Encryption Standard) vznikl na popud soutěže ministerstva obchodu USA v roce 1973. Jeho základ vyvinula firma IBM zdokonalením vlastního systému LUCIFER, který do té doby IBM používala pro své potřeby, po úpravách byl patentován 24. 2. 1975, od března byl zveřejněn pro použití. Je určen pro ochranu neutajovaných dat v civilním sektoru. Předpokládalo se, že bude používán 10-15 let, v daném čase se však za něj nenašla dostatečná náhrada, a proto byl používán ve finančním sektoru i v dalších letech. Přestože jeho narušení již bylo dokumentováno a je k dispozici kompletní dokumentace k dešifrovacímu stroji, používá se v řadě aplikací dodnes.

runda

vícerundový systém

DES patří do třídy blokových šifer [33, 49]. Otevřený text (OT) je rozdělen na bloky po 64 bitech. Každý z těchto bloků je jako celek zpracován na šifrovaný text (ŠT) také o délce 64 bitů s použitím klíče K, viz obr. 16. Délka klíče je 56 bitů, vzniká ze slova délky 8 Byte vynecháním jeho paritních bitů. Zpracování každého bloku otevřeného textu probíhá v 16 opakováních (***rundách***). V každém z nich je z klíče K vybráno podle určitého postupu 48 bitů tvořících pracovní klíč K_i viz obr. 17.

Vlastní blokový algoritmus při šifrování zpracovává blok otevřeného textu tak, že ho nejprve podrobí počáteční permutaci IP (permutuje 64 bitů) a poté rozdělí na pravou (R_0) a levou (L_0) polovinu o 32 bitech. Nyní probíhá 16 opakovaných kroků, kdy z dvojice (L_i, R_i) je za účasti klíče K_i vytvořena dvojice (L_{i+1}, R_{i+1}). R_i je nejprve rozšířen z 32 na 48 bitů pomocí expanze E a k výsledku je binárně (bit po bitu) přičten klíč K_i. Výsledek K_i XOR $E(R_i)$ je rozdělen do osmi částí (po 6 bitech), které procházejí přes substituční boxy S1 ÷ S8. Tyto S-boxy jsou v podstatě pamětí ROM 6x4 bity a výstupem je pak 8 x 4 = 32 bitů, realizují nelineární funkci vstupních bitů. Výstup je upraven permutací P (32 na 32 bitů), označuje se obvykle $f(R_i, K_i)$, neboť je funkcí klíče K_i a slova R_i. Poslední operace v kroku jsou operace $L_{i+1} = R_i$, $R_{i+1} = L_i$ XOR $f(R_i, K_i)$. Po proběhnutí 16 opakování je provedena záměna L_{16} a R_{16} a 64 bitový blok (R_{16}, L_{16}) je permutován permutací IP^{-1} (permutace inverzní k IP), čímž obdržíme šifrovaný text.



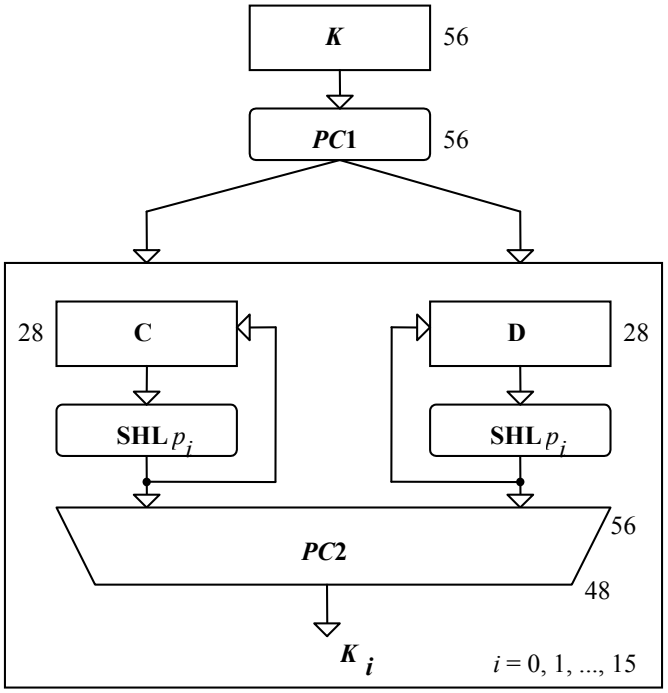
Obr. 16. Blokové schéma DES

Tvorba klíčů K_i probíhá podle obr. 17. Klíč K o 56 bitech je podroben permutaci $PC1$ a naplněn do dvou 28-bitových registrů C a D. Obsah registrů je v každém kroku posunut doleva o p_i bitů (posun je v krocích i = 0, 1, 8, 15 jednobitový, jinak dvoubitový) a jejich výsledné 56-bitové zřetězení je podrobeno permutačnímu výběru $PC2$, který kromě permutace 56 bitů provede redukci na 48 bitů vynecháním některých bitů. Výstupem je klíč K_i.

Feistelův systém

Výhodou algoritmu je, že při šifrování a dešifrování může být použito stejné technické zařízení (DES zpočátku nebylo povoleno aplikovat v softwarové podobě). Pouze pořadí výběru klíčů je obrácené, místo K_1, K_2, ..., K_{16} se používá K_{16}, K_{15}, ..., K_1. Při vytváření klíčů se navíc operace SHL[#] nahradí operací SHR a jinou tabulkou posunů p_i. Tento systém zpracování se nazývá ***Feistelův*** podle dr. Horsta Feistela, kryptologa IBM a vynálezce systému LUCIFER.

V poslední době se objevuje řada připomínek k nízké bezpečnosti systému. Zejména omezení délky bloku na 64 bitů je považováno za nevýhodné. Díky rozvoji procesorů schopných rychle šifrovat v DES je možné nalezení klíče v reálném čase postupem vyzkoušení všech možných klíčů, což je nejtriviálnější postup dešifrování. Podrobné návody na sestrojení dešifrovacího stroje v ceně pod dva miliony dolarů jsou dnes volně k dispozici na Internetu.



Obr. 17. Tvorba klíčů K_i v DES

[#] Operace SHL představuje rotaci bitů směrem doleva (podle způsobu obvyklého zápisu tedy významově nižší bity postupují na vyšší pozici), nejlevější bit je přenesen na nejpravější pozici (tzv. rotace s přenosem). Operace SHR představuje obdobnou rotaci opačným směrem.

12.2 Systém RC4 – Rivest Cipher

systém RC4

Tvůrcem RC4 je Ronald Rivest z RSA, RC4 je jednou z nejpoužívanějších proudových šifer pro Internet a komerční využití. Po celých 7 let se RSA dařilo utajit algoritmus RC4. Poté byla dissasemblována hackerem a její popis umístěn na Internetu. RSA dostala strach o zneužití jejich šifry konkurencí, jelikož nebyla patentována. Tato šifra je řádově desetkrát rychlejší než DES. RC4 je využita v SSL 3.0 společnosti Netscape, v Microsoft Office, ORACLE Secure SQL nebo v Microsoft Windows 2000. Klíč pro RC4 může mít maximálně 256 Byte (2048 bitů). Pro funkčnost se uvažuje o klíčích zarovnaných na Byte. V praxi se využívá 128bitový klíč na území USA a 40bitový mimo. Vstupem je klíč o volitelné délce. Klíč dále inicializuje konečný automat generující proud hesla, které se XORuje na otevřený text. Princip šifry je míchání Byte klíče spojené s permutací klíče. ¨

RC4 je velmi zajímavá, neobyčejná a analytickou metodou zatím nenapadnutá šifra. Ani teoretický základ šifry není doposud řádně prozkoumán.

systém RC5

Druhý algoritmus z dílny Ronalda Rivesta z roku 1994 je RC5. RC5 přinesl do kryptografie novou myšlenku o použití rotací závislých na datech. Jedná se o velmi pružný algoritmus s mnoha parametry. Šifrovací klíč má 0-255 Byte, počet rund šifrovacího procesu (0-255) a délka slova z hodnot 16, 32, 64, 128 a 256 Byte, přičemž algoritmus zpracovává bloky o dvojnásobné délce slova. Pro své použití lze algoritmus přizpůsobit vhodnou volbou parametrů. Délka slova 128 bitů se používá pro realizaci ***výtahu zprávy*** (***hash***) pro elektronický podpis, jak bude popsán dále. Zvětšování počtu rund vede ke zvyšování bezpečnosti algoritmu na úkor rychlosti. Šest rund postačí pro nenáročné aplikace, 32 pro ty nejnáročnější. Jako rozumná se jeví volba délka slova 32 bitů, 12 rund, 16 Byte klíče, což krátce zapíšeme RC5-32/12/16.

výtah zprávy hash

systém RC6

Algoritmus RC6 je vylepšená verze RC5, která tímto dosahuje požadavků NIST na nový standard AES. Byly přidány některé funkce, celočíselné násobení v klíči a čtyři pracovní registry místo dvou. Parametrizovaný je stejně jako RC5, leč v soutěži nezvítězil, viz další kapitola.

12.3 Systém AES

systém AES

Šifrovací systém AES (Advanced Encryption Standard) vznikl jako náhrada již nevyhovujícího systému DES opět po složitém procesu výběrového řízení. Je standardizován jako Federal Information Processing Standard (FIPS) pro použití ve státních organizacích U.S.A.

Veřejná soutěž byla vyhlášena dne 2. 1. 1997 National Institute of Standards and Technology (NIST – http://www.nist.gov/) a ukončena 2. 10. 2000. Vítězem se stal systém Rijndael (čti: rájndol), pocházející z Belgie. Jeho autory jsou **Dr. Joan Daemen** (Yo'-ahn Dah'-mun) a **Dr. Vincent Rijmen** (Rye'-mun). Předpokládá se použití systému pro období 2000 – 2030.

Jedná se o blokovou šifru, délka bloku 128 bitů, délka klíče 128, 192 nebo 256 bitů (4, 6, 8 32bitových slov), vícerundová, počet rund 10, 12 a 14 závisí na

délce klíče. Pracuje se s prvky Galoisova tělesa GF(2^8) a s polynomy, jejichž koeficienty jsou prvky z tohoto tělesa. Prvky v Galoisově tělese mají osm bitů, reprezentují polynomy $b_7x^7 + b_6x^6 + \ldots + b_0$. Základní operace jsou realizovány v okruhu polynomů modulo $m(x) = x^8 + x^4 + x^3 + x^1 + 1$ [**8**].

12.3.1 Postup šifrování AES

1. Před šifrováním se vypočítá $4 + 4.N_r$ rundovních klíčů (32bitových slov). První 4 se naxorují na otevřený text („whitening“) a umístí po sloupcích do matice **A**.

$$\mathbf{A} = \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix}$$

2. Proběhne N_r rund, v každé se použijí 4 rundovní klíče. Jedna runda probíhá

```
Round (State, RoundKey)
{
ByteSub (State)
ShiftRow (State)
MixColumn (State)                    kromě poslední rundy
AddRoundKey (State, RoundKey)
}
```

ByteSub
bytová substituce $a \rightarrow b$:
vypočítá se multiplikativní inverze prvku a: $c = a^{-1} \bmod m(x)$, poté se Byte c transformuje na b substitucí:

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} + \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{bmatrix}$$

ShiftRow
cyklická rotace prvků matice **A** v jednotlivých řádcích doleva, první řádek beze změny, druhý o jeden prvek, třetí o dva atd.

MixColumn
zesložitění prvků v rámci každého sloupce matice **A**:

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

např. $b_0 = '02'.a_0 \oplus '03'.a_1 \oplus '01'.a_2 \oplus '01'.a_1$

AddRoundKey
Naxorování znaků (Byte) klíče na prvky matice **A** po sloupcích.

12.3.2 Zpracování klíče AES

Šifrovací klíč k o N_k 32bitových slovech (4, 6 nebo 8) se naplní na počátek pomocného pole **W**[0 … N_{k-1}]. Toto pole následně expanduje:

```
for i = Nk to 4*Nr + 3 do
{
    temp = W[i-1];
    if (i mod Nk = 0)
        temp = SubByte(RotByte(temp)) ⊕ Const[i/Nk];
    if ((i mod Nk = 4) AND (Nk = 8))
        temp = SubByte(temp);
    W[i] = W [i – Nk] ⊕ temp;
}
```

Využívají se operace:
RotByte – cyklický posun bytů doprava, **SubByte** – substituce bytů stejná jako u šifrování, aplikovaná na všechny byty proměnné **temp** a pole konstant **Const**[].

Kontrolní otázky:
43. Jaký je princip realizace proudové šifry?
44. Jaká jsou rizika při opakovaném použití klíče proudové šifry?
45. Jaké principy používají současné blokové systémy s tajným klíčem?
46. Charakterizujte systém AES.

Shrnutí obsahu kapitoly
V této kapitole jsme se seznámili se současnými kryptografickými systémy s tajným klíčem. Nejprve byla prezentována proudová šifra, která je označována za jediný nerozluštitelný kryptografický systém. Dále jsou prezentovány různé blokové systémy RC4 až RC6, DES a AES, dnešní standardy.

13 Systémy s veřejným klíčem

Cíl:

Cílem celé kapitoly je seznámení se současnými kryptografickými systémy s veřejným klíčem a ukázán systém Diffie-Hellman a RSA včetně ukázky generování dvojice klíčů. Jsou vysvětleny požadavky na elektronický podpis zpráv a ukázány principy jeho realizace. Na závěr jsou popsány požadavky na konstrukci hašovací funkce, která je nezbytná pro realizaci elektronického podpisu.

Po jejím prostudování byste měli být schopni:

- Vysvětlit princip fungování systémů s veřejným klíčem a realizaci elektronického podpisu zpráv.
- Ukázat jak se generují klíče v systému s tajným klíčem RSA.
- Popsat požadavky na hašovací funkci a způsoby její realizace.

Klíčová slova této kapitoly:

Binární složitost, polynomiální složitost, systém s veřejným klíčem, jednosměrná funkce, elektronický podpis, autentifikace, identifikace, výtah zprávy, otisk zprávy, hašovací funkce, systém Diffie-Hellman, systém RSA, metoda kvadratického síta, metoda číselného tělesa, PGP, systém IDEA, bezkoliznost, kolize, orákulum, náhodné orákulum, zarovnání, Damgard-Merkelovo zesílení, kontext, inicializační hodnota, Davies-Meyerova konstrukce, kompresní funkce, runda, MD5, SHA-2.

Doba potřebná ke studiu: 4 hodiny

Průvodce studiem

Studium této kapitoly je nutné k pochopení funkce systémů s veřejným klíčem, jejich princip a realizaci včetně ukázky činnosti systému RSA a generování dvojice klíčů. jsou popsány požadavky na elektronický podpis a ukázán postup jeho realizace včetně požadavků na hašovací funkci a postupu její realizace.
Na studium této části si vyhraďte čtyři hodiny. Vyzkoušejte si také realizaci systému s veřejným klíčem a generování klíčů.

Moderní kryptografické systémy (od 80. let 20. století) se od klasických systémů liší zásadně tím, že systém šifrování i použitý klíč jsou zveřejněny. Samotná znalost klíče pro šifrování totiž není dostatečná k určení dešifrovacího klíče v reálném čase.

K posouzení této skutečnosti se pracuje s ***binární složitostí*** operací. Za bitovou operaci přitom považujeme sčítání dvou jednobitových čísel (ať již s přenosem nebo bez přenosu).

binární složitost

13.1 Binární bitová složitost

Sčítání dvou k-bitových čísel má binární bitovou složitost úměrnou délce k. Například sečtení následujících dvou čísel má binární složitost 8:

```
  10101100        172
+ 11101010      + 234
————————————    ———————
 110010110        406
```

Podobně je tomu u jiných operací. Násobení dvou k a j bitových čísel představuje v nejhorším případě sčítání j sčítanců, po doplnění nulami dlouhých nejvýše $(k + j - 1)$ bitů. Pro dosažení výsledku musíme sčítat první s druhým, výsledek se třetím sčítancem atd. Tedy celkem nejvýše $(j - 1)$ součtů $(k + j - 1)$ místných čísel. Pro odhad binární bitové složitosti pak zjednodušíme:
$(j - 1)\cdot(k + j - 1) < j\cdot(k + j) < 2\cdot k\cdot j$.

Binární bitová složitost násobení je tedy úměrná $2\cdot k\cdot j$.

```
    101101      45 · 13 = 585
      1110
————————————
    101101
   101101
  101101
————————————
  1001001001
```

Pro desítkovou soustavu platí, že dekadické číslo n bude mít nejvýše k binárních znaků, kde je $k \leq \log_2(n) + 1$. Sčítání dvou takových čísel má tedy binární bitovou složitost úměrnou hodnotě $\log_2(n) + 1$.

Násobení dekadických čísel n a m má binární bitovou složitost úměrnou: $2\cdot k\cdot j = 2\cdot[\log_2(n) + 1]\cdot[\log_2(m) + 1] = 2\cdot[\log_2(n)\cdot\log_2(m) + \log_2(n\cdot m) + 1] < 4\cdot \log_2(n)\cdot\log_2(m)$.

Binární bitovou složitost budeme zapisovat zkráceně: $f(n) = O(g(n))$, jestliže pro funkce f a g definované na množině přirozených čísel existuje konečná limita:

$$\lim_{n\to\infty} \frac{f(n)}{g(n)}. \tag{90}$$

Můžeme tedy zapsat:

1. dekadické číslo n má $O(\log_2(n))$ binárních číslic,
2. součet dvou dekadických čísel n reprezentuje nejvýše $O(\log_2(n))$ bitových operací,
3. součin (podíl) dekadických čísel n a m vyžaduje $O(\log_2(n)\cdot\log_2(m))$ bitových operací.

polynomiální složitost

Pojem ***polynomiální složitost*** budeme používat u operací, u kterých výpočet funkce $f(n)$ dekadického čísla n, které má k dvojkových cifer, vyžaduje

$$O(k^d) \text{ bitových operací,} \tag{91}$$

kde je d – přirozené číslo.

Operace + – · / tedy mají polynomiální složitost,
pro + a – je $d = 1$,
pro · a / je $d = 2$.

Výpočet čísla $n! = 1\cdot2\cdot....\cdot n$ má při použití postupného násobení binární bitovou složitost

$$O(n\cdot\log_2^2(n)) = O(2^k\cdot k^2). \tag{92}$$

Tento algoritmus tedy nemá ***polynomiální složitost***. To mimo jiné znamená, že jeho složitost s rostoucím n roste rychleji.

Známý Eukleidův algoritmus pro nalezení největšího společného dělitele čísel (a, b) má bitovou složitost $O(\log_2^2(a))$ pro $a > b$. Můžeme ho zapsat např.:

```
FUNCTION GCD(a, b: INTEGER): INTEGER;
VAR x: INTEGER;
BEGIN
        a:= ABS(a); b:= ABS(b);
        WHILE a > 0 DO
            BEGIN
                x:= a;
                a:= b MOD a;
                b:= x;
            END;
        GCD:= b;
END;
```

Stejnou bitovou složitost má nalezení čísel u, v splňujících rovnost: $u\cdot a + v\cdot b = (a, b)$ nebo algoritmus na zjištění nesoudělnosti čísel, když je $(a, b) = 1$.

13.2 Principy systémů s veřejným klíčem

systém s veřejným klíčem

Klasické šifrovací postupy můžeme zapsat vždy jako uspořádanou čtveřici *(K, M, C, T*), kde jsou *K, M, C* konečné množiny klíčů, zpráv a šifrovaných textů, T: K x $M \rightarrow C$ je zobrazení, které každému klíči $k \in K$ přiřadí prosté zobrazení $T_k = T(k, .)$.

Např. pro afinní šifru a telegrafní abecedu $N = 26$, kdy T_k: $y = (a \cdot x + b) \bmod 26$ popíšeme kryptografický systém:

$K = \{(a, b); (a, 26) = 1, a, b \in Z_{26}\}$ – prostor klíčů,
$M = Z_{26} = \{0, 1, ..., 25\}$ – prostor zpráv,
$C = Z_{26}$ – prostor šifrovaných zpráv.

Pro určení počtu různých klíčů potřebujeme funkci $\Phi(n)$, která udává počet čísel menších a nesoudělných s argumentem n (přirozené číslo). Pokud známe faktorizaci čísla n (rozklad na prvočinitele), tedy $n = p_1^{e_1} \cdot p_2^{e_2} \cdot ... \cdot p_k^{e_k}$, pak platí:

$$\Phi(n) = n\left(1 - \frac{1}{p_1}\right)\left(1 - \frac{1}{p_2}\right)...\left(1 - \frac{1}{p_k}\right). \qquad (93)$$

Prostor klíčů má pak pro afinní šifru $\Phi(26) \cdot 26 = 312$ prvků, neboť $26 = 2^1 \cdot 13^1$, tedy

$$\Phi(26) = 26\left(1 - \frac{1}{2}\right)\left(1 - \frac{1}{13}\right) = 26 \cdot 0{,}5 \cdot 0{,}92 = 12\,. \qquad (94)$$

Bitová složitost výpočtu dešifrovacího klíče je $O(\log_2^3 n)$, má tedy polynomiální složitost. To znamená, že je v reálném čase řešitelný. Až do roku 1976 se věřilo, že je tomu tak u každého kryptografického systému. Tehdy byly nalezeny tzv. ***jednosměrné funkce*** (One-Way Function), u kterých je jednoduché vypočítat její funkční hodnotu $y = f(x)$ pro každé

jednosměrná funkce

$x \in D_f$,
kde je D_f – definiční obor funkce f,

ale velmi těžké vypočítat hodnotu $x = f^{-1}(y)$ pro skoro všechna

$y \in H_f$,
kde je H_f – obor hodnot funkce f.

Možnost vyřešení všech možných hodnot inverzní funkce je přitom nutno eliminovat velkou mohutností této množiny. Za jednoduché přitom považujeme algoritmy, které mají polynomiální složitost.

K otázkám narušitelnosti těchto systémů je třeba přistupovat velmi obezřetně, neboť dosud nebyla nalezena prakticky upotřebitelná funkce, která by byla prokazatelně jednosměrná.

13.3 Architektura systému s veřejným klíčem

V klasickém kryptografickém systému pro n účastníků posílal každý účastník každému zprávy šifrované podle jiného klíče. Bylo tedy potřeba

$$\binom{n}{2} = \frac{n \cdot (n-1)}{2} \tag{95}$$

různých klíčů, které je navíc potřeba často měnit, aby se předešlo jejich odhalení.

jednosměrná funkce

V systému s veřejným klíčem může kdokoliv poslat adresátovi šifrovanou zprávu, kterou dokáže dešifrovat jen adresát. K tomu se používá právě jednosměrných funkcí. Každý účastník zveřejní svoji adresu a šifrovací klíč formou jakéhosi telefonního seznamu. Odesílatel zašifruje zprávu podle tohoto klíče a odešle. Dešifrovat ji dokáže jen adresát. Než si takový systém blíže ukážeme, zamysleme se nad problémy ověření přijaté zprávy. Příjemce musí být schopen ověřit, zda přijatá zpráva pochází skutečně od uvedeného odesílatele. Zda nemá zpráva za úkol poškodit adresáta. Např. v bankovním styku příkazem k úhradě z jeho konta apod. Nad tím se zamýšlí tzv. problém ***elektronického podpisu***.

elektronický podpis

Problém podpisu zprávy má v zásadě tři roviny. Jsou to

autentifikace
identifikace

***Autentifikace*:** A může dokázat B, že je A,
narušitel nemůže dokázat B, že je A.

***Identifikace*:** A může dokázat B, že je A,
B nemůže dokázat nikomu, že je A (B tedy nemůže být narušitelem).

***Podpis*:** A může dokázat B, že je A,
B nemůže dokázat ani sám sobě, že je A (B nedokáže napodobit podpis A).

Uveďme některá možná řešení problému podpisu:

- Odesílatel A odešle zprávu adresátu B, zašifrovanou podle klíče k_B z telefonního seznamu,

výtah zprávy
otisk zprávy

- Na závěr zprávy uvede ***výtah zprávy*** nebo také ***otisk zprávy*** (message digest) – označme ho z, který zašifruje pomocí svého dešifrovacího klíče k_{A-1} a vyšle jako zprávu y, Pro výtah zprávy se používají různé ***hašovací funkce*** (hash function), tedy jednodměrné funkce, jednak proto, aby nebylo možno ze samotného haše vypočítat zprávu a také proto, aby dvě různé zprávy měly velmi odlišné haše. Podrobněji bude problematika našívacích funkcí později.

hašovací funkce

  Celkem tedy vyšle:
  $c = T(k_B, C)$,
  $y = T(k_B, \mathrm{T}(k_{A-1}, z)) = T(k_B, T^{-1}(k_A, z))$.

Adresát B přijme zprávu c a vypočítá výtah zprávy, ten ověří spolu s dešifrovaným výtahem, který dešifruje nejprve svým dešifrovacím klíčem a pak šifrovacím klíčem odesílatele:

$$z' = T(k_A, T(k_{B-1}, z)).$$

Jestliže se vypočítaná a přijatá hodnota výtahu zprávy rovnají, pak tuto zprávu mohl odeslat jen odesílatel A, neboť jen on je schopen v reálném čase vypočítat svůj dešifrovací klíč.

13.4 Systém Diffie-Hellman

systém Diffie-Hellman

Systém Diffie-Hellman byl svými autory publikován na podzim roku 1975 jako způsob, jak si dva účastníci komunikace mohou vyměnit šifrovací klíče bez nebezpečí jejich získání třetí stranou. Pracuje následujícím způsobem:

1. Oba účastníci si zvolí dvě čísla α a q. A to zcela veřejně.
2. Každý účastník si zvolí tajné číslo X_i a odešle partnerovi Y_i jako výsledek operace:
   $Y_i = \alpha^{X_i} \bmod q$. Kde $i = 1, 2$ pro jednotlivé účastníky.
3. Po výměně vypočítají společný (sdílený) klíč K_{12}:
   $K_{12} = Y_2^{X_1} \bmod q = \alpha^{X_1 X_2} \bmod q$, pro prvního účastníka a
   $K_{12} = Y_1^{X_2} \bmod q = \alpha^{X_1 X_2} \bmod q$, pro druhého účastníka.

Tímto způsobem se oba účastníci bezpečným způsobem dohodli na klíči systému s tajným klíčem, aniž by si přitom vyměnili svá tajně zvolená čísla X. Algoritmus Diffie-Hellman je velmi jednoduchý a používá se v řadě aplikací, včetně programu PGP, který bude uveden dále.

13.5 Systém RSA

systém RSA

Věnujme se nyní blíže alespoň stručnému popisu konkrétního systému s veřejným klíčem. Podrobné informace lze nalézt v literatuře, např. [19, 33].

Systém RSA vytvořili autoři Rivest, Shamir, Adleman v r. 1977 v Massachusetts Institute of Technology. Využili starý problém nalezení faktorizace velkých čísel (rozkladu na prvočinitele). Zatímco faktorizace malých čísel (řádově do 256 000) je běžně dostupná, pro velká čísla (řádově 100-místná) je v reálném čase neřešitelná. Na tomto principu postavili následující postup.

Postup při RSA šifrování:

1. Převedeme alfanumerické vyjádření znaků na numerické (obvykle se používá ASCII kód).
2. Text rozdělíme na bloky stejné délky. Obsah jednoho bloku vyjádříme jako x.
3. Zvolíme číslo N, které je součinem dvou náhodně zvolených 100-místných prvočísel $N = p \cdot q$. Číslo N má tedy 200 míst v dekadickém vyjádření. Přitom chceme, aby platilo $1 \le x < N$.

4. Zvolíme šifrovací exponent s, tak aby byl nesoudělný s $\Phi(N)$, tedy aby platilo
   $(s, \Phi(N)) = (s, (p-1)(q-1)) = 1$.
5. Každý účastník zveřejní čísla N a s spolu se svou adresou v telefonním seznamu.
6. Dále najde číslo t, aby vyhovovalo kongruenci $t \cdot s \equiv 1 \bmod \Phi(N)$, respektive $t \cdot s \equiv 1 \bmod (p-1)(q-1)$. Protože platí $(s, \Phi(N)) = 1$, má tato kongruence jediné řešení.
7. K šifrování bude použita transformace $y = x^s \bmod N$ a k dešifraci transformace $x = y^t \bmod N$.
8. Šifrovaný text se přenáší běžným přenosovým kanálem.

Bezpečnost sytému je dána tím, že pro určení exponentu t potřebujeme znát prvočísla p, q, která nebyla zveřejněna a pro velké N je jich tolik, že nalezení správných prvočísel je v reálném čase nemožné.

Příklad 12.2: Zašifrujme zkratku P R F O S U
15 17 05 14 18 20
délku bloku volíme 3, $p = 31$, $q = 37 \Rightarrow N = p \cdot q = 1147$, $\Phi(N) = (p-1)(q-1) = 1080 = 2^3 \cdot 3^3 \cdot 5$, volíme nesoudělné $s = 7$. Dešifrovací exponent je řešením $t \cdot 7 = 1 \bmod 1080$, $t = 463$.

Šifrujeme:
$151^7 \bmod 1147 = 263$,
$705^7 \bmod 1147 = 091$,
$141^7 \bmod 1147 = 632$,
$820^7 \bmod 1147 = 1104$.

Šifrovaný text zní: 803875001917

Dešifrujeme:
$263^{463} \bmod 1147 = 151$,
$091^{463} \bmod 1147 = 705$ atd.

Pokud se čtenáři jeví umocňování na 463 nerealizovatelné, postačí si uvědomit následující skutečnost, platnou pro násobení v modulo N:

$$x \cdot x \bmod N = (x_1 \cdot N + x_2)(x_1 \cdot N + x_2) \bmod N =$$
$$= (x_1^2 \cdot N + 2 \cdot x_1 \cdot x_2 \cdot N + x_2^2) \bmod N = x_2 \cdot x_2 \bmod N \qquad (96)$$

Operaci $x^s \bmod N$ tedy můžeme realizovat postupným násobením v modulo N:

$$((x \cdot x \bmod N) \cdot x \bmod N) \ldots \qquad (97)$$

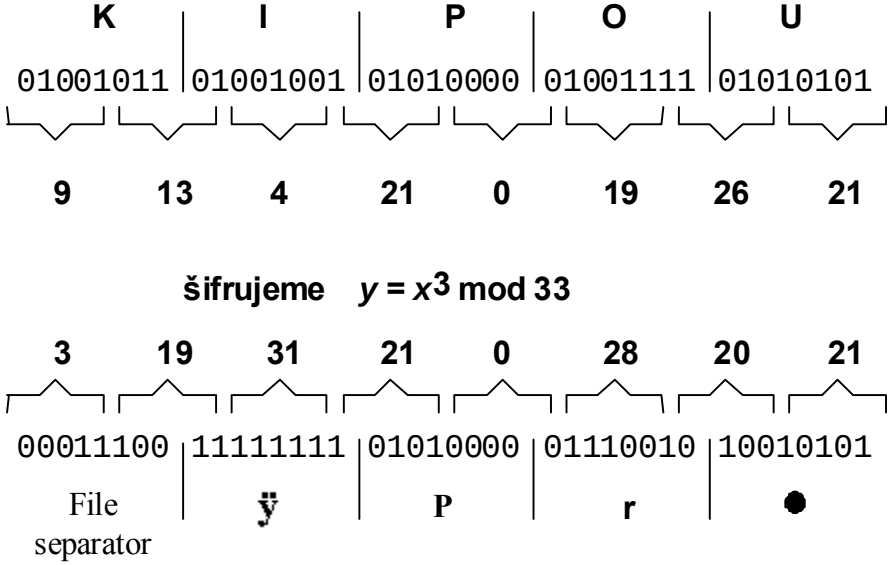
Pozorný čtenář si jistě při rozboru řešeného příkladu uvědomil dva problémy. Za prvé skutečnost, že např. $820^7 \bmod 1147 = 1104$, tedy číslo, které nelze vyjádřit pomocí tří číslic, a tedy ani přenést zvoleným systémem. Tento problém v řešeném příkladu vystupuje pouze proto, že příklad je ilustrační. Ve skutečnosti musíme zvolit takovou délku bloku, aby bylo možno v něm vyjádřit všechna čísla menší než N. Zejména při práci s binárními čísly to zjevně nebude velký problém.

Druhý problém v ilustračním příkladu nevystupuje. Může se stát, že po zašifrování obdržíme opět původní blok Např. při šifrování bloku 001 obdržíme opět blok 001, tedy nezašifrovaný blok. U tohoto bloku nás to nemůže překvapit, ani u bloku 000. Vzniká však otázka, kolik existuje různých bloků, které nebudou zašifrovány při použití daného klíče. V našem případě je jich celkem 49. Pro jiné klíče může být toto číslo vyšší, u některých klíčů dokonce nebude zašifrován ani jeden blok! Takové klíče označujeme jako ***poloslabé*** a ***slabé klíče***. Tabulka 12 uvádí rozbor klíčů RSA pro malá prvočísla.

Příklad 12.3: Máme za úkol vytvořit RSA systém pro šifrování pětibitových bloků dat. Zvolíme $p = 3$, $q = 11 \Rightarrow N = 33$, $\Phi(N) = 2.10 = 2^2 \cdot 5$. Zvolíme nesoudělný šifrovací exponent $s = 3$, dešifrovací exponent se pak rovná $t = 7$. Blok pěti bitů nabývá hodnot $x \in \langle 0, 31 \rangle$. Pro šifrování platí:

$x \to y$	$x \to y$	$x \to y$	$x \to y$	$x \to y$	$x \to y$	$x \to y$	$x \to y$
$0 \to 0$	$4 \to 31$	$8 \to 17$	$12 \to 12$	$16 \to 4$	$20 \to 14$	$24 \to 30$	$28 \to 7$
$1 \to 1$	$5 \to 26$	$9 \to 3$	$13 \to 19$	$17 \to 29$	$21 \to 21$	$25 \to 16$	$29 \to 2$
$2 \to 8$	$6 \to 18$	$10 \to 10$	$14 \to 5$	$18 \to 24$	$22 \to 22$	$26 \to 20$	$30 \to 6$
$3 \to 27$	$7 \to 13$	$11 \to 11$	$15 \to 9$	$19 \to 28$	$23 \to 23$	$27 \to 15$	$31 \to 25$

text „KIPOU“ vyjádříme v kódu ASCII, a rozdělíme na bloky délky 5 bitů:



I tento příklad je nutné chápat jako ilustrační, pro praktické použití je velikost bloku příliš malá.

Bezpečnost systému RSA je založena na obtížnosti faktorizace velkých čísel. Zatím nebyl nalezen jiný útok na systém než rozklad čísla N na původní prvočísla p a q. Získat takový rozklad je ale velmi těžké a časově náročné. S rozšiřováním použití algoritmu RSA se samozřejmě intenzivně pracuje na metodách faktorizace velkých čísel. Jde především o ***metodu kvadratického síta*** – QS (Quadratic Sieve) a ***metodu číselného tělesa*** – NFS (Generalized Number Field Sieve). NFS je novější a zdá se, že pro čísla nad 130 cifer i rychlejší. Současně je vhodné poznamenat, že doposud nebylo prokázáno, že jediný útok je možný faktorizací čísla N, tedy je možné, že existuje algoritmus, který rozluští zašifrovaný text jinak, než touto cestou.

metoda kvadratického síta

metoda číselného tělesa

V květnu roku 1999 rozvířil kryptoanalytické hladiny jeden z autorů RSA, izraelský matematik Adi Shamir. Představil totiž zařízení TWINKLE, které umožňuje mnohonásobně zrychlit metody QS nebo NFS. TWINKLE je zkratkou z názvu „The Weizmann INstitute Key Locating Engine". TWINKLE je optoelektrické zařízení, které umožňuje masivně paralelním způsobem urychlit metody QS a NFS. Toto zařízení způsobilo posun velikostí používaných klíčů na dvojnásobek, tedy velikosti od 1024 bitů, které mohou být stále považovány za bezpečné. Bohužel se pro elektronický obchod stále používají implementace RSA s klíčem 512 bitů, což již rozhodně nelze považovat za bezpečnou délku [76].

PGP

Patrně nejznámějším použitím RSA je jeho aplikace v systému ***PGP*** (Pretty Good Privacy), běžně používaném v počítačové síti Internet [17]. Protože realizace RSA je přece jen časově náročná, šetří se čas následujcím postupem. Vlastní zpráva se zašifruje tajným klíčem jednorázového použití, který vygeneruje odesílatel (používá se systém ***IDEA*** – International Data Encryption Standard, který je považován za silnější než DES). Příjemce tento klíč dostane zašifrovaný svým veřejným klíčem RSA systému. Protože zabezpečení celého systému je závislé na kvalitě použitého šifrovacího algoritmu RSA nebo DH/DSS, nabízí systém tři úrovně ochrany:

systém IDEA

- běžný (1024 bitů) – lze jej rozluštit s vynaložením značného úsilí,
- obchodní (2048 bitů) – možná jej dokáží luštit státní orgány,
- vojenský (4096 bitů) – všeobecně považován za nerozluštitelný.

Kromě toho systém PGP řeší také problematiku digitálního podpisu a autentifikace předávaných zpráv, kompresi a kompatibilitu s elektronickou poštou. Vytvořil ho Philip Zimmermann.

Pro realizaci RSA algoritmu jsou dnes dostupné realizační procesory, takže není nutné provádět potřebné výpočty programově. Samostatným problémem je však určení prvočíselnosti zvolených velkých čísel (řádově 100-místných).

Postupujeme tak, že generátorem pseudonáhodných čísel vygenerujeme 100-místné číslo m, pokud je sudé, pracujeme s číslem $m + 1$. Určení jeho prvočíselnosti pomocí Eukleidova algoritmu je časově nereálné, proto se používají tzv. pravděpodobnostní algoritmy určení prvočíselnosti, i ty jsou běžně dostupné.

Nejjednodušší postup je určení, zda m je prvočíslo na k pokusů. (ukázka neřeší problém operací se 100-místnými čísly, ani realizaci funkce RANDOM, která je použita!)

```
PROGRAM PRVOCISELNOST;
VAR m, k, a, b, i: LONGINT;
BEGIN
    READ(m, k);
    FOR i:= 1 to k do
        BEGIN
            a:= RANDOM(1, m - 1); {funkce vrátí
                                   náhodné číslo
                                   v daném rozsahu}
            b:= ( a ** (m - 1) MOD m);
            IF b <>1 THEN
                BEGIN
                    WRITELN(m,' je slozene cislo');
                    EXIT;
                END;
        END;
    WRITELN(m, ' je asi prvocislo');
END.
```

Pro dostatečně velké *k* je pravděpodobnost úspěchu dosti velká. Existuje však řada zlepšení algoritmu, např. [19, 72].

Tab. 12. Určení počtu shodných zpráv pro vybrané klíče RSA

p	*q*	Φ(*n*)	*N*	*s*	dešif. exp.	nazašif. bloky
3	5	8	15	3	3	9
3	**5**	**8**	**15**	**5**	**5**	**15**
3	5	8	15	7	7	9
3	7	12	21	5	5	9
3	**7**	**12**	**21**	**7**	**7**	**21**
3	7	12	21	11	11	9
3	11	20	33	3	7	9
3	11	20	33	9	9	9
3	**11**	**20**	**33**	**11**	**11**	**33**
3	11	20	33	13	17	9
3	11	20	33	19	19	9
3	13	24	39	5	5	15
3	13	24	39	7	7	21
3	13	24	39	11	11	9
3	**13**	**24**	**39**	**13**	**13**	**39**
3	13	24	39	17	17	15
3	13	24	39	19	19	21
3	13	24	39	23	23	9
5	7	24	35	5	5	15
5	7	24	35	7	7	21
5	7	24	35	11	11	9
5	**7**	**24**	**35**	**13**	**13**	**35**
5	7	24	35	17	17	15
5	7	24	35	19	19	21
5	7	24	35	23	23	9
5	11	40	55	3	27	9
5	11	40	55	7	23	9
5	11	40	55	9	9	15
5	11	40	55	11	11	33
5	11	40	55	13	37	15
5	11	40	55	17	33	15
5	11	40	55	19	19	9
5	**11**	**40**	**55**	**21**	**21**	**55**
5	11	40	55	29	29	15
5	11	40	55	31	31	33
5	11	40	55	39	39	9
31	37	1080	1147	7	463	49
31	37	1080	1147	31	871	217
31	37	1080	1147	37	613	259

13.6 Hašovací funkce

hašovací funkce

Hašovací funkce mají v současných informačních a komunikačních systémech několik významných úkolů. Zpočátku byla jako hašovací funkce označována funkce, která libovolně velkému vstupu přiřazovala krátký ***hašovací kód*** o pevně určené délce. V současné době se označení hašovací funkce používá v kryptografii pro ***kryptografickou hašovací funkci***, která má oproti původní definici ještě navíc vlastnosti ***jednosměrnosti (one-way)*** a ***bezkoliznosti (collission-free)***.

jednosměrná funkce

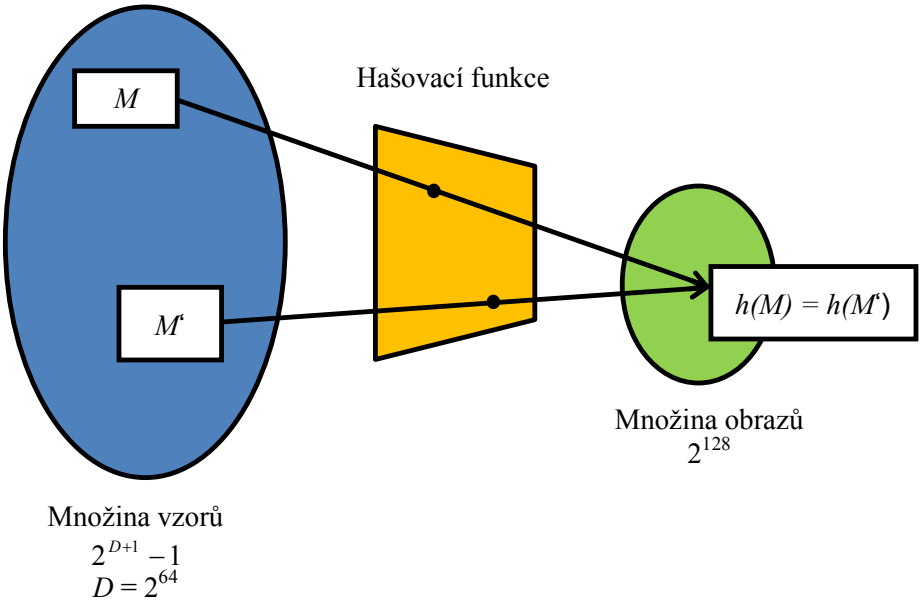
bezkoliznost

Jednosměrnost znamená, že z M je možno snadno vypočítat $h(M)$, ale obráceně to je pro náhodně zadaný hašovací kód H výpočetně nezvládnutelné.
Bezkoliznost prvního řádu (nebo také ***odolnost proti kolizi***) vyžaduje, aby bylo výpočetně nezvládnutelné nalezení libovolných dvou různých (třebas naprosto nesmyslných) zpráv M a M' tak, že $h(M) = h(M')$. Pokud se toto stane, říkáme, že jsme nalezli **kolizi**. Taková bezkoliznosti, je nazývána bezkoliznost prvního řádu nebo jednoduše bezkoliznost. Taková vlastnost je důležitá u elektronických podpisů. Umožňuje nám podepisovat pouze hašovací kód (zkráceně haš) a nikoliv celou zprávu, která může být značně dlouhá (u MD5/SHA-1/SHA-256 prakticky až do délky $D = 22^{64} - 1$ bitů).

kolize

Bezkoliznost druhého řádu nebo také odolnost proti nalezení druhého vzoru požaduje, aby pro daný náhodný vzor x bylo výpočetně nezvládnutelné nalezení druhého vzoru $y \neq x$ pro které platí $h(x) = h(y)$.

Při zkoumání bezkoliznosti je třeba si uvědomit, že existuje velmi mnoho různých zpráv $(1 + 2^1 + ... + 2^D = 2^{D+1} - 1)$ pro maximální délku zprávy D. Naproti tomu existuje jen málo hašovacích kódů (u algoritmu MD-5 je to jen 2^{128}). Nutně tedy existuje velké množství kolizí (v průměru je to kolem 2^{D-127}). Jde tedy o to, aby nebyly nalezitelné v dostupném čase.



Obr. 18. Kolize při použití hašovací funkce

13.6.1 Náhodné orákulum

orákulum

náhodné orákulum

Jako ***orákulum*** označujeme libovolný stroj, který na vstup odpovídá nějakým výstupem, a to na stejný vstup vždy stejným výstupem (obvykle se jedná o myšlenkový experiment, který neumíme sestrojit). Jako ***náhodné orákulum*** pak označujeme orákulum, které novému vstupu přiřadí náhodnou hodnotu výstupu a zapamatuje si ji pro příští použití. Jakmile je orákulu předložen stejný vstup, odpoví opět stejným výstupem. Je tedy zřejmé, že bychom byli rádi, kdyby se hašovací funkce chovala jako náhodné orákulum. Hašovací funkce pak může plnit řadu úloh, jednak u elektronických podpisů, ale také generování šifrovacích klíčů na základě jednodušších hesel, ukládání otisků hesel namísto hesel samotných, ale také např. v konstrukci generátorů pseudonáhodných čísel. V takovém případě by složitost nalezení vzoru (zprávy) ke známému obrazu (hašovacímu kódu) byla rovna 2^n , kde n – počet bitů obrazu.

Protože víme, že kolize existují, ptáme se, jak velká musí být množina náhodných zpráv, aby se v ní s významnou pravděpodobností vyskytovaly alespoň dvě zprávy se stejným hašovacím kódem. Pro 50% pravděpodobnost výskytu kolize bychom očekávali, že pro n-bitovou hašovací funkci to bude množina $\frac{1}{2}2^n$ vzorů. Známý narozeninový paradox však říká, že pro n-bitovou hašovací funkci nastává kolize se zhruba 50% pravděpodobností již v množině $2^{\frac{n}{2}}$ vzorů. Například pro 160bitový hašovací kód bychom očekávali $\frac{1}{2}2^{160} \doteq 7{,}31.10^{47}$, ale bude to pouze $2^{80} \doteq 1{,}21.10^{24}$. To způsobuje, že dostupné hašovací funkce jsou náchylné na kolize prvního druhu (najdeme nějaké dvě různé zprávy se stejným hašovacím kódem), ale nalezení kolize druhého druhu je daleko náročnější, protože musíme najít kolizi pro přesně určenou zprávu.

13.6.2 Konstrukce hašovacích funkcí

Zpráva zpracovávaná hašovací funkcí může být velmi dlouhá (např. $D = 2^{64} - 1$), takže ji musíme zpracovávat po blocích. K tomu je také nutné zarovnat zprávu na délku danou celistvým násobkem délky bloku. Toto ***zarovnání*** přitom musí být jednoznačně odejmutelné. Nemůže to například být pouze řada nul, protože by nebylo zřejmé, zda zpráva sama nekončí nulou (nebo více nulami). Nejčastěji je proto zarovnání definováno jako jednička a řada nul, za kterými následuje délka původní zprávy. Toto rozšíření se nazývá ***Damgard-Merklovo zesílení***, protože bylo navrženo nezávisle oběma autory na konferenci Crypto 1989.

zarovnání

Damgard-Merklovo zesílení

Například pro velikost bloku m = 512, který používají hašovací funkce MD4, MD5, SHA-0, SHA-1, SHA-256 se doplnění se provádí tak, že M je nejprve doplněna bitem 1, poté co nejmenším počtem nulových bitů (může jich být 0 ÷ 447) tak, aby do celistvého násobku 512 bitů zbývalo ještě 64 bitů, a nakonec je těchto 64 bitů vyplněno 64bitovým vyjádřením počtu bitů původní zprávy

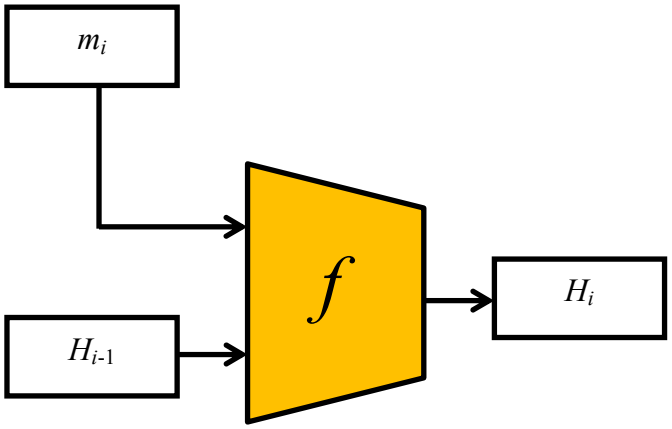
M. Délka zprávy pak také vstupuje do hašovacího procesu. Rezervace 64 bitů na délku zprávy umožňuje hašovat zprávy až do délky $D = 2^{64} - 1$ bitů.

Všechny současné prakticky používané hašovací funkce používají ***Damgard-Merklův*** (DM) princip iterativní hašovací funkce s využitím kompresní funkce. Kompresní funkce f zpracovává blok m_i zprávy a produkuje výsledek, označovaný jako ***kontext*** H_i postupně pro $i = 1, 2, \ldots, N$. Hodnota kontextu je pak vstupem do dalšího iteračního kroku. Na počátku musí být kontext H_0 nastaven na ***inicializační hodnotu*** (IV – initial value), která je určena jako konstanta v definici hašovací funkce.

kontext

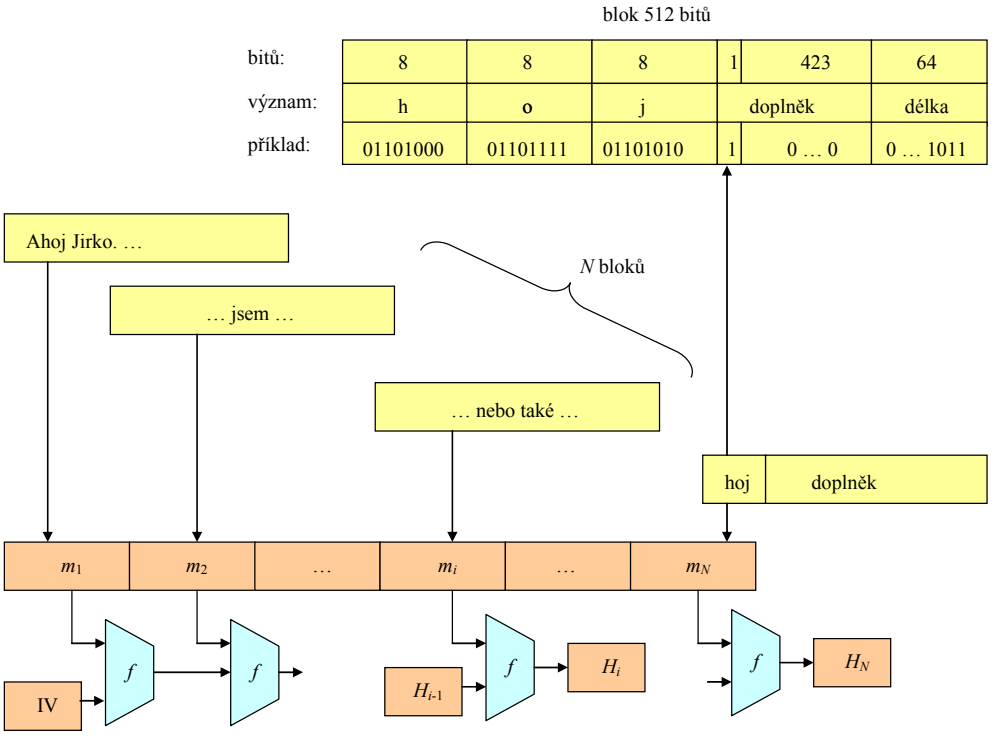
inicializační hodnota

Kontextem bývá obvykle několik 16bitových nebo 32bitových slov, u MD5 jsou to čtyři slova A, B, C, D (celkem 128 bitů). Po zpracování posledního bloku m_N získáme kontext H_N, z něhož bereme buď celou délku, nebo jeho část jako výsledný hašovací kód. U funkce MD5 je šířka kontextu 128 bitů a výsledným hašovacím kódem je všech 128 bitů kontextu H_N.



Obr. 19. Damgard-Merklův princip iterativní hašovací funkce s využitím kompresní funkce

Pokud je hašovací funkce bezkolizní, bude bezkolizní také kompresní funkce. Opačně je situace složitější. Nicméně pro Damgard-Merklovu konstrukci hašovací funkce bylo dokázáno, že pokud je bezkolizní kompresní funkce, je bezkolizní také hašovací funkce.



Obr. 20. Princip doplňování, kompresní funkce a iterativní hašovací funkce

13.6.3 Konstrukce kompresní funkce

Jak bylo ukázáno, pro kvalitní hašovací funkci potřebuje zkonstruovat kvalitní kompresní funkci *f*. Ta musí být velmi robustní, musí zajistit dobré promíchání bitů zprávy a jednosměrnost. Můžeme je konstruovat např. na základě známých jednosměrných funkcí a použít znalosti z oblasti blokových šifer:

$$H_i = E_{m_i}(H_{i-1})$$

kde je E – kvalitní bloková šifra.

Davies-Meyerova konstrukce kompresní funkce

Davies-Meyerova konstrukce kompresní funkce pak zesiluje vlastnost jednosměrnosti ještě přičtením vzoru před výstupem:

$$H_i = f(H_{i-1}, m_i) = E_{m_i}(H_{i-1}) \text{ XOR } H_{i-1}$$

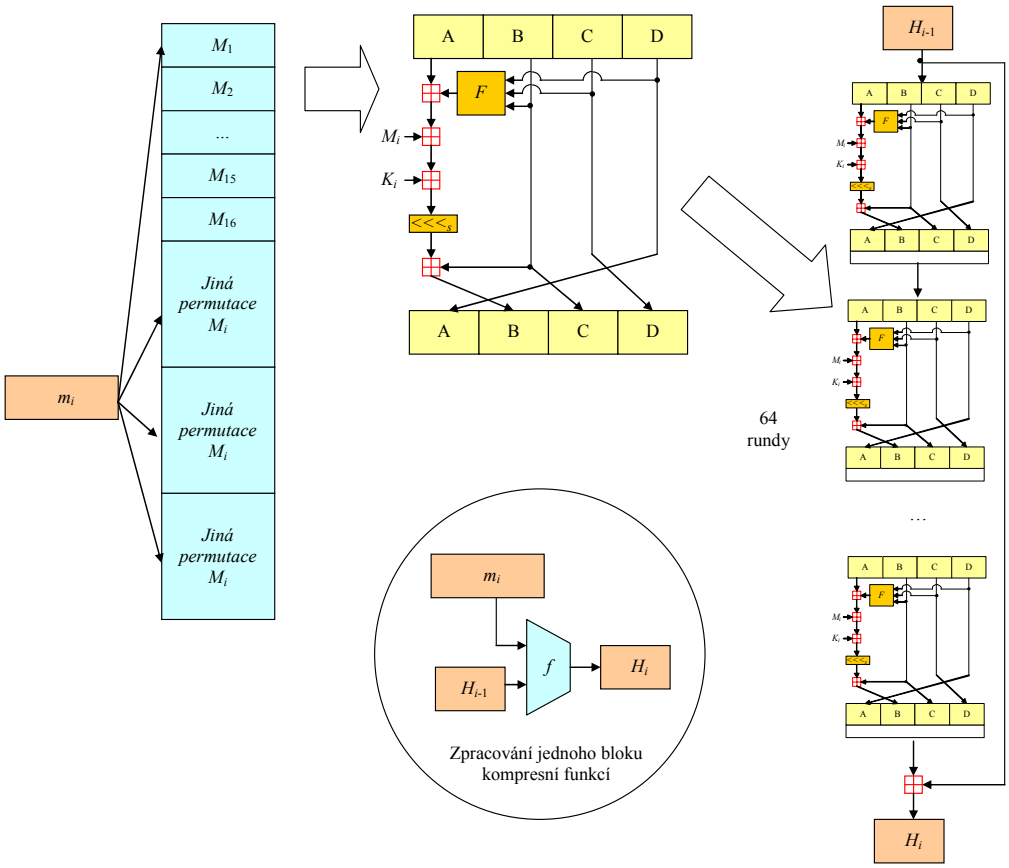
runda

Výstup je zde tedy navíc maskován vstupem, což ještě více ztěžuje případný zpětný chod. Protože blok zprávy mi je obvykle velmi velký (512 bitů) a klíče blokových šifer tak dlouhé nebývají, aplikuje se bloková šifra několikrát za sebou (v ***rundách***), přičemž se (rundovní) klíč postupně čerpá z bloku m_i.

13.6.4 Kompresní a hašovací funkce MD5

MD5

U algoritmu MD5 tvoří kontext čtyři 32bitová slova A, B, C a D. Na obrázku vidíme zvětšenu jednu rundu hašování, kde m_i je jeden 512bitový blok zprávy. Ten je rozdělen na šestnáct 32bitových slov M_0, M_1, ..., M_{15}, a tato posloupnost je opakována čtyřikrát za sebou (v různých permutacích). Na obrázku vidíme, že v kompresní funkci se kontext zašifruje vždy jedním 32bitovým slovem M_i. Doplňme, že na místě dílčí funkce F v obrázku se po 16 rundách střídají čtyři různé (nelineární i lineární) funkce (F, G, H, I) a v každé rundě se využívá jiná konstanta K_i. Po 64 rundách dojde ještě k přičtení původního kontextu (H_{i-1}) k výsledku podle Davies-Meyerovy konstrukce (funkce XOR je nahrazena aritmetickým součtem modulo 2^{32}). Tak vznikne nový kontext H_i. Pokud by zpráva M měla jen jeden blok, byl by kontext (A, B, C, D) celkovým výsledkem. Pokud ne, pokračuje se stejným způsobem v hašování druhého bloku zprávy m_2 jakoby s inicializační hodnotou H_{i-1}. Po zpracování bloku m_N zůstane v registrech výsledný 128bitový hašovací kód H_N.



Obr. 21. Princip algoritmu MD5

Vzhledem k rozšíření algoritmu MD5 se samozřejmě okamžitě stal předmětem rozsáhlých útoků. Bylo nalezeno několik postupů nalezení kolizí prvního řádu. To znamená, že útočník umí najít dvě různé (náhodné) zprávy se stejným hašovacím kódem. Ale není schopen stejně snadno nalézt kolizi druhého řádu, tedy najít druhý vzor (zprávu) se stejným hašovacím kódem.

Podobně byly prezentovány útoky schopné se složitostí 2^{33} hašovacích operací pokořit algoritmus SHA-0 a se složitostí 2^{69} hašovacích operací algoritmus SHA-1 (narozeninový paradox očekává 2^{80} operací). To je vzhledem k současnému výkonu počítačů sice stále dost velké číslo, ale vzhledem ke stálému růstu výkonu počítačů (v souladu s Mooreovým zákonem) nelze tuto skutečnost podceňovat. Jsou dokumentovány postupy, jak může útočník oklamat certifikační autoritu a získat certifikát na dva různé šifrovací klíče [41].

13.6.5 Třída hašovacích funkcí SHA-2

SHA-2

Z důvodu zvýšení odolnosti proti kolizím je od 1. února 2003 k dispozici nová trojice hašovacích funkcí (Secure Hash Algorithm) SHA-256, SHA-384 a SHA-512 (souhrnně označované jako SHA-2) a od února 2004 SHA-224 (dodatek SHA-2). Tyto funkce zvětšuji délky hašovacího kódu na 256, 384 a 512 bitů (SHA-224 má 224 bitový hašovací kód), což odpovídá složitosti 2^{128}, 2^{192} a 2^{256} pro nalezení kolizí narozeninovým paradoxem. To je na jedné straně dost vysoká složitost a také to odpovídá složitosti útoku hrubou silou na tři délky klíčů, které nabízí šifrovací standard AES. Pokud se týká konstrukce nových funkcí, jsou velmi podobné SHA-1 a používají stejné principy, ale pracují se složitějšími funkcemi a širšími vstupy. Jejich cílem bylo poskytnout větší odolnost proti kolizi a nabídnout odpovídající bezpečnost jako klíče pro AES.

Vývoj v této oblasti však pokračuje velmi rychle, např. v roce 2012 byla po pěti letech ukončena soutěž pro nový standard SHA-3 a přitom byl ve stejné době vydán další standard [40], který používá již známý standard SHA512, s výhodou využívající 64bitové operace na 64bitových procesorech. Jeho výstup je pak zkrácen na potřebnou délku *t*. Tak vznikají algoritmy SHA-512/*t*, neboli SHA-512/224, SHA-512/256, SHA-512/384 a SHA-512/512.

Kontrolní otázky:

47. Jak hodnotíme složitost funkce?
48. Jaké jsou požadavky na jednosměrnou funkci, uveďte její příklad?
49. Jaký je princip systému RSA, ukažte na příkladu jeho použití?
50. Jaké známe typy kolizí u našívacích funkcí, bezkoliznost vůči které z nich umíme zajistit?

Korespondenční úkol:

6. Zvolte si dvě dvojciferná prvočísla a s jejich pomocí ukažte konstrukci šifrovacího a dešifrovacího klíče systému RSA. Ukažte jejich použití jak pro šifrování, tak dešifrování.

Shrnutí obsahu kapitoly

V této kapitole jsme se seznámili se současnými systémy s veřejným klíčem. Byl vysvětlen přístup k hodnocení složitosti funkcí a pojem jednocestná funkce. S jejich pomocí pak byly definovány příklady systémů s veřejným klíčem a související problém elektronického podpisu. Samostatně byly popsány požadavky na hašovací funkci a její realizace pro zajištění správného výtahu zprávy pro realizaci elektronického podpisu.

14 Řešení úkolů

1. Z četností výskytu zpráv určíme pravděpodobnosti jejich výskytu, následně jejich informační obsah. Součet součinů pravděpodobností a informačních obsahů zpráv dává informační entropii zdroje:

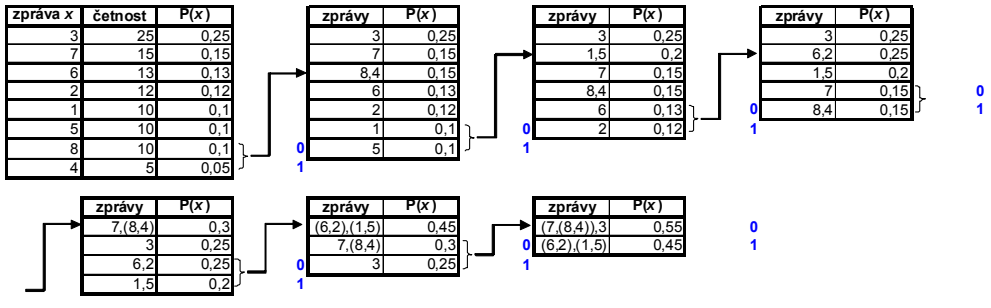
zpráva x	četnost	$P(x)$	$k(x)$	$P(x)k(x)$
1	10	0,1	3,32	0,33
2	12	0,12	3,06	0,37
3	25	0,25	2,00	0,50
4	5	0,05	4,32	0,22
5	10	0,1	3,32	0,33
6	13	0,13	2,94	0,38
7	15	0,15	2,74	0,41
8	10	0,1	3,32	0,33
	100		**H =**	**2,87**

2. Vezmeme si na pomoc průměrný informační obsah zprávy a vynásobíme jím počet přenesených zpráv, tím získáme přenosovou rychlost kanálu:

počet zpráv za sekundu	650	[zpráv.s^{-1}]
průměrný informační obsah	2,87	[b]
přenosová rychlost	1865,5	[b.s^{-1}]

3. K_1: Kraftova nerovnost 1,375, není jednoznačně dekódovatelný.
   K_2: Kraftova nerovnost 1,111, není jednoznačně dekódovatelný.
   K_3: Kraftova nerovnost 0,815, je jednoznačně dekódovatelný i prefixový.
   K_4: Kraftova nerovnost 0,666, je jednoznačně dekódovatelný není prefixový.

4. Podle předpisu pro Huffmanovu konstrukci kódu provádíme postupné redukce abecedy, následně označíme každé rozhodování a určíme kódová slova. Následně určíme jednotlivá kódová slova a součet součinů pravděpodobnosti výskytu zprávy a délky kódového slova určí střední délku kódového slova. Z ní a známé informační entropie zdroje určíme redundanci daného způsobu kódování:



zpráva x	četnost	$P(x)$
3	25	0,25
7	15	0,15
6	13	0,13
2	12	0,12
1	10	0,1
5	10	0,1
8	10	0,1
4	5	0,05

zprávy	$P(x)$
3	0,25
7	0,15
8,4	0,15
6	0,13
2	0,12
1	0,1
5	0,1

zprávy	$P(x)$
3	0,25
1,5	0,2
7	0,15
8,4	0,15
6	0,13
2	0,12

zprávy	$P(x)$
3	0,25
6,2	0,25
1,5	0,2
7	0,15
8,4	0,15

zprávy	$P(x)$
7,(8,4)	0,3
3	0,25
6,2	0,25
1,5	0,2

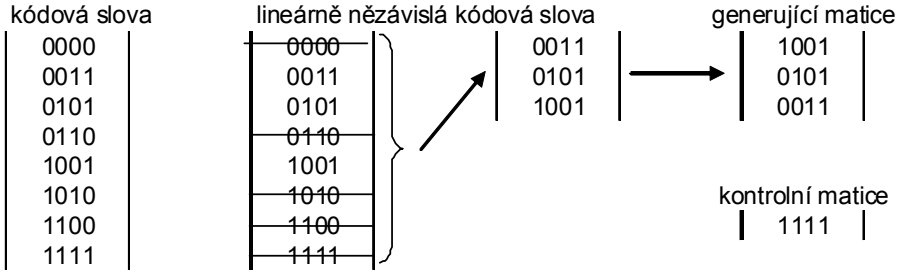
zprávy	$P(x)$
(6,2),(1,5)	0,45
7,(8,4)	0,3
3	0,25

zprávy	$P(x)$
(7,(8,4)),3	0,55
(6,2),(1,5)	0,45

zpráva x	kód	$N(x)$	$P(x)$	$P(x)N(x)$
1	110	3	0,10	0,30
2	101	3	0,12	0,36
3	01	2	0,25	0,50
4	0011	4	0,05	0,20
5	111	3	0,10	0,30
6	100	3	0,13	0,39
7	000	3	0,15	0,45
8	0010	4	0,10	0,40
			L =	**2,90**
			R = L - H =	**0,03**

5. Hammingovy vzdálenost d = 2, kód celkové kontroly parity objeví jednoduché chyby, neopraví žádnou chybu.

6. Z kódových slov určíme lineárně nezávislá kódová slova a z nich sestavíme generující matici. Protože pořadí kódových slov v generující matici může být libovolný, uspořádáme je tak, aby tvořily jednotkovou matici. Z ní určíme kontrolní matici daného kódu.



7. Standardní dekódovací tabulka

	chyb. slovo							
třída 1. (kód)	0000	0011	0101	0110	1001	1010	1100	1111
třída 2.	1000	1011	1101	1110	0001	0010	0100	0111

Tabulka pro dekódování pomocí syndromů:

e_i	$s_{e_i}^T$
0000	0
1000	1

$s_{1011} = 1, s_{1110} = 1, s_{1111} = 0$, potom dekódování probíhá následovně:

$\partial(1011) = (1011) - (1000) = (0011)$,

$\partial(1110) = (1110) - (1000) = (0110)$,

$\partial(1111) = (1111) - (0000) = (1111)$.

24. Protože se jedná o (7, 4)-kód, představuje informační slovo 1001 polynom u(z) = $z^6 + z^3$. Po dělení informačního slova generujícím polynomem g(z) = $z^3 + z + 1$ získáme zbytek $r(z) = z^2 + z$. Celé kódové slovo tedy bude v(z) = $z^6 + z^3 + z^2 + z$ (1001110).

15 Rejstřík

R

Ř

S

Š

T

V

Z

Vysvětlivky k používaným symbolům

Průvodce studiem – vstup autora do textu, specifický způsob, kterým se studentem komunikuje, povzbuzuje jej, doplňuje text o další informace

Příklad – objasnění nebo konkretizování problematiky na příkladu ze života, z praxe, ze společenské reality, apod.

Pojmy k zapamatování.

Shrnutí – shrnutí předcházející látky, shrnutí kapitoly.

Literatura – použitá ve studijním materiálu, pro doplnění a rozšíření poznatků.

Kontrolní otázky a úkoly – prověřují, do jaké míry studující text a problematiku pochopil, zapamatoval si podstatné a důležité informace a zda je dokáže aplikovat při řešení problémů.

Úkoly k textu – je potřeba je splnit neprodleně, neboť pomáhají dobrému zvládnutí následující látky.

Korespondenční úkoly – při jejich plnění postupuje studující podle pokynů s notnou dávkou vlastní iniciativy. Úkoly se průběžně evidují a hodnotí v průběhu celého kurzu.

Úkoly k zamyšlení.

Část pro zájemce – přináší látku a úkoly rozšiřující úroveň základního kurzu. Pasáže a úkoly jsou dobrovolné.

Testy a otázky – ke kterým řešení, odpovědi a výsledky studující najdou v rámci studijní opory.

Řešení a odpovědi – vážou se na konkrétní úkoly, zadání a testy.

16 Literatura

[1] ADÁMEK, J. *Kódování*. Praha: SNTL 1989. 192 s.

[2] BAHENSKÝ, Z. *Huffmanův kód*. Elektronika, 1990, č. 1, s. 27-28.

[3] BOSÁKOVÁ, D., KUČEROVÁ, A., PECA J, & VONDRUŠKA, P. *Elektronický podpis – přehled právní úpravy, komentář k prováděcí vyhlášce k zákonu o elektronickém podpisu a výklad základních pojmů*. Olomouc: Nakladatelství ANAG, 2002. ISBN 80-7263-125-X.

[4] BŘEZINA, T. a kol. *Informatika pro strojní inženýry* I. Brno: ES VUT Brno 1991, 187 s.

[5] CHEN, C. L. Byte-Oriented Error-Correcting Codes for Semiconductor Memory Systems. *IEEE Transactions on Computers*, vol. 35, no. 7, pp. 646-648, July 1986, doi:10.1109/TC.1986.1676807.

[6] CORGE, CH. *Elementy informatyki. Informatyka i myśl ludska*. Warszawa: PW Naukowe 1981. (překlad z francouštiny)

[7] ČASTOVÁ, N. & ŠARMANOVÁ, J. *Počítače a algoritmizace*. Ostrava: VŠB Ostrava 1983.

[8] DAEMEN, J. & RIJMEN. V. *AES Proposal: Rijndael*. Document version 2, Date 03/09/99. NIST.

[9] *Data Protection Codes of Practice*. Oxford (UK): Blackwell Publischers 1990. 126 s.

[10] DECROS a. s. *Web information system* [on-line]. HTML Format [cit. 2002-10-31]. Dostupný z: < URL: http://www.decros.cz/>

[11] DOBDA, L. *Ochrana dat v informačních systémech*. Praha: Grada, 1998. 286 s. ISBN 80-7169-479-7.

[12] DOBEŠ, J. & ŽALUD, V. *Moderní radiotechnika*. Praha: BEN – technická literatura, 2006,768 s. ISBN 80-7300-132-2.

[13] FALK, B. *Průvodce světem internetu*. 1. vyd. Praha: Computer Press 1994, 310 s.

[14] FARANA, R. *Kapitoly ze základů informatiky*. 1. vyd. Ostrava: VŠB-TU Ostrava, 2003. 108 s. ISBN 80-248-0265-1.

[15] FARANA, R. Systémy ochrany dat založené na generátoru náhodných čísel. In *Proceedings of International Scientific Conference '95 Session No 20 Automation of Technological Processes*. Ostrava: VŠB–TU Ostrava 1995, s. 5/1 - 5/6.

[16] FARANA, R. *Šifrování pro radost a poučení*. Brno: Mravenec 1994, 60 s. ISBN 80-85978-00-8.

[17] GARFINKEL, S. *PGP: Pretty good privacy. Šifrování pro každého.* 1. vyd. Praha: Computer Press, 1998. 373 s. ISBN 80-7226-054-5.

[18] GLADMAN, B. *AES Algorithm Efficiency* [on-line].]. HTML Format [cit. 2002-10-31]. Dostupný z: <URL: http://fp.gladman.plus.com/ cryptography_technology/aes/index.htm>.

[19] GROŠEK, O. & PORUBSKÝ, Š. *Šifrovanie. Algoritmy, metódy, prax*. Praha: Grada 1992.

[20] *Handbook of Data Communications*. Oxford: Blackwell Publischers 1993.

[21] HÁTLE, J. & KAHOUNOVÁ, J. *Úvod do teorie pravděpodobnosti*. Praha: SNTL 1987.

[22] HLAVIČKA, J., RACEK, S., GOLAN, P. & BLAŽEK, T. *Číslicové systémy odolné proti poruchám*. 1. vyd. Praha: Vydavatelství ČVUT, 1992, 330 s. ISBN 80-01-00852-5.
[23] HORČÍK, R. Rostislav Horčík's homepage [on-line]. Last update: Feb 25, 2013, HTML formát [cit. 2013-08-10]. Dostupný z: <URL: http://www2.cs.cas.cz/~horcik/>.
[24] HRDA, S. Crypt. *Elektronika*, 1993, č. 4/93, s. 30-31.
[25] HYNEK, J. *Komprese dat Huffmanovým kódováním*. PC World, č. 8, 1994, s. 110- 112.
[26] *Informační bezpečnost. Příručka manažera*. Praha: Tate International, s. r. o., 2001. 130 s. ISBN 80-902858-4-8.
[27] IVÁNEK, J. *Základy kódování a kryptografie*. Ostrava: Ostravská univerzita v Ostravě, 2006, 73 s. Studijní materiál pro distanční studium.
[28] JANEČEK, J. *Gentlemani (ne)čtou cizí dopisy*. 1. vyd. Brno: Books, 1998, 176 s. ISBN 80-85914-90-5.
[29] JANEČEK, J. *Odhalená tajemství šifrovacích klíčů minulosti*. 1. vyd. Praha: Naše vojsko, 1994, 184 s. ISBN 80-206-0462-6.
[30] JANEČEK, J. *Rozluštěná tajemství*. Praha: Petr Tychlt – NAKLADATELSTVÍ XYZ, 2008, 268 s. ISBN 978-80-86864-96-9.
[31] JANOUŠEK, R., KLIMEŠ, C. *Kódování*. Ostrava: Ostravská univerzita v Ostravě, 2005, 127 s. Učební text pro distanční studium.
[32] JIRÁČEK, M. Elektronické peníze a dokumenty. *Sdělovací technika*, r. 41, 1993, č. 3, s. 94-95.
[33] JIRÁČEK, M. Utajení dat. *Sdělovací technika*, r. 41, 1993, č. 2, s. 55-59.
[34] KAČMÁŘ, D., FARANA, R. *Vybrané algoritmy zpracování informací*. 1. vyd. Ostrava: VŠB-TU Ostrava, 1996. 136 s. ISBN 80-7078-398-2.
[35] KALUŽA, J., KALUŽOVÁ, L. & MAŇASOVÁ, Š. *Informatika*. 5. vyd. Ostrava: VŠB–TU Ostrava, 2000, 152 s. ISBN 80-7078-394-X.
[36] KALUŽA, J., KALUŽOVÁ, L. & MAŇASOVÁ, Š. *Základy informatiky v ekonomice*. Ostrava: VŠB Ostrava 1992.
[37] KARÁSEK, J. Hashovací funkce MD5 a kolize souborů se stejným hashem. *Elektrorevue*. [on-line]. 2010, č. 50, s. 50-1 – 50-6 [cit. 2012-11-08]. ISSN 1213-1539. Dostupné z <http://www.elektrorevue.cz/cz/download/hashovaci-funkce-md5-a-kolize-souboru-se-stejnym-hashem/>.
[38] KELEMEN, J. AJ. *Základy umelej inteligencie*. Bratislava, Alfa 1992.
[39] KLIKA, O. & LÉBL, M. *Teorie diskrétních kódů*. 1. vyd. Praha: NADAS 1968, 192 s.
[40] KLÍMA, V. Digitální otisk SHA-4 přeskočil SHA-3. *Sdělovací technika – elektronika – multimédia* [on-line]. 2012, č. 5, s. 2-3 [cit. 2012-11-08]. Dostupné z <cryptography.hyperlink.cz/2012/ST_2012_05_18_19.pdf>.
[41] KLÍMA, V. *Hašovací funkce, principy, příklady a kolize* [on-line]. verze 1, 19. 3. 2005. [cit. 2012-11-08]. Dostupné z <http://crypto-world.info/klima/2005/cryptofest_2005.htm>.
[42] KLÍMA, V. Kódování, kódy komprimace a šifrování. *Chip*, 1993, č. 2, s. 24-28.
[43] KLÍMA, V. Kolaps šifrového standardu. *Chip*, 1993, č. 5., s. 52-58.
[44] KLÍMA, V. Metody zabezpečení dat. *Chip*, č. 8, 1993, s. 166-169.
[45] KLÍMA, V. Metody zabezpečení dat - 2. *Chip*, č. 9, 1993, s. 212-215.

[46] KLÍMA, V. Nebezpečné softwarové šifry. *Softwarové noviny*, 1995, č. 6, s. 120- 124.
[47] KLÍMA, V. Programátorské šifry a jejich luštění. *Softwarové noviny*, 1995, č. 3, s. 104-107.
[48] KLÍMA, V. *Skrytá válka šifer. Význam ochrany dat šifrováním*. 1. vyd. DECROS, 1995, 56 s.
[49] KLÍMA, V. Šifrování. Algoritmus DES. *Chip*, 1993, č. 5, s. 52-58, č. 6, s. 50-55, č. 7, s. 50-51.
[50] KLÍMA, V. Šifrování. Šifry, právo a tajné služby. *Chip*, 1993, č. 12, s. 228-229, r. 1993, č. 1, s. 146-149.
[51] KLÍMA, V. Utajené komunikace - díl 1-8. *Chip*, 1994, č. 5-12.
[52] KLÍMA, V. Utajené komunikace - díl 9-19. *Chip*, 1995, č. 1-11.
[53] KNUDSEN, L. R. *The Block Cipher Lounge – AES* [on-line]. HTML Format [cit. 2002-10-31]. Dostupný z: < URL: http://www.ii.uib.no/~larsr/aes.html>
[54] KONEČNÁ, P. *Kódy a šifry – jejich matematický základ a historie*. Ostrava: Ostravská univerzita v Ostravě, 2006, 83 s.
[55] KÖNIGOVÁ, M. AJ. *Matematické a statistické metody v informatice*. 1. vyd. Praha: Státní pedagogické nakladatelství, 1988, 189 s.
[56] KOŽNAR, Z. & KOŽNAROVÁ, M. *Tieňohra*. (Překlad z češtiny). Bratislava: MON 1988, s. 98-101.
[57] KUCHAŘ, M. *Bezpečná síť. Jak zajistíte bezpečnost Vaší sítě*. Brno: UNIS Publishing, 1999. 92 s. ISBN 80-7169-886-5.
[58] KUCHAŘ, M. *Bezpečná síť.* Praha: Grada Publishing, spol. s r. o., 1999, 92 s. ISBN 80-7169-886-5.
[59] KULIKOWSKI, J. L. *Informacja i świat w którym żyjemy*. Warszawa: PW „Wiedza Powszechna“ 1978.
[60] LIN, SHU & COSTELLO DANIEL J. *Error Control Cosiny*. Prentice Hall; 2004. ISBN 978-0130426727.
[61] MACKAY, DAVID J.C. *Information Theory, Inference, and Learning Algorithms*. Cambridge, Cambridge University Press, 2003, 640 s.
[62] MERTA, A. & MERTOVÁ, D. *Anglicko-český slovník informatiky*. 1. vyd. Praha: Ústředí vědeckých, technických a ekonomických informací, 1988, 215 s. ISBN 80-212-0000-6.
[63] MOLNÁR, Z. *Moderní metody řízení informačních systémů*. Praha, Grada 1992.
[64] MOON, TODD K. *Error Correction Coding: Mathematical Methods and Algorithms*. New Jersey: Wiley-Interscience, 2005. ISBN 978-0471648000.
[65] MOOS, P. *Informační technologie*. 1. vyd. Praha: Vydavatelství ČVUT, 1993, 200 s. ISBN 80-01-01048-1.
[66] OLŠÁK, P. *BI - Lineární algebra (pro FIT)* [on-line]. HTML formát [cit:2013-08-12]. Dostupný z: <URL: http://petr.olsak.net/bilin/>.
[67] PŘIBYL, J. & KODL, J. *Ochrana dat v informatice*.Praha: Vydavatelství ČVUT, 1996, 299 s. ISBN 80-01-01664-1.
[68] RODRYČOVÁ, D. & STAŠA, P. *Bezpečnost informací jako podmínka prosperity firmy*. 1. vyd. Praha: Grada Publishing, 2000. 143 s. ISBN 80-7169-144-5
[69] RSA SECURITY. *Web information system* [on-line]. HTML Format [cit. 2002-10-31]. Dostupný z: < URL: http://www.rsasecurity.com/>

[70] RYAN, WILLIAM & LIN, SHU. *Channel Codes: Classisal and Modern*. Cambridge: Cambridge University Press, 2009. ISBN 978-0521848688.
[71] SALAVEC, K. Test prvočíselnosti. *Bajt*, 1991, č. 5/91, s. 64.
[73] SINGH, S. *Kniha kódů a šifer – Tajná komunikace od Starého Egypta po kvantovou kryptografii.* Překlad P. Koubský a D. Eckhardtová. Praha: Nakladatelství Dokořán a Argo, 2003, 382 s. ISBN 80-86569-18-7 (Dokořán), 80-7203-499-5 (Argo).
[74] SKLAR, B. *Reed-Solomon codes*. [online], 2011. HTML format [cit. 2011-05-21]. Dostupný z: <URL: http://ptgmedia.pearsoncmg.com/images/art_sklar7_reed-solomon/elementLinks/art_sklar7_reed-solomon.pdf>.
[75] SKUTIL, P. Něco pro Sira A. C. Doyla. *Bajt*, 1992, č. 1/92, s. 101.
[76] SOUČEK, J & BENEŠ, T. *Matematické principy informační bezpečnosti 2000/2001*. [on-line]. HTML Format [cit. 2002-10-Dostupný z: <URL: http://www.mujweb.cz/veda/gcucmp/Mff/seminar.html>.
[77] STAUDEK, J. & MOTYČKOVÁ, L. *2x15 kapitol z distribuovaných systémů. Část I Distribuované algoritmy a počítačové sítě*. Brno, VUT 1993.
[78] STAUDEK, J. Bezpečnost počítačových sítí. *Computer echo*, 1992, č. 2/92, s. 4-10.
[79] TĚŠITELOVÁ, M. AJ. *O češtině v číslech*. Praha: Academia 1987.
[80] TURSKI, W. M. *Propedeutyka informatyki*. Warszawa, PW Naukowe 1989.
[81] UHLÍŘ, J. & SOVKA, P. *Číslicové zpracování signálů*. 1. vyd. Praha: Vydavatelství ČVUT, 1995, 313 s. ISBN 80-01-01303-0.
[82] VLČEK, J. *Inženýrská informatika*. 1. vyd. Praha: Vydavatelství ČVUT, 1994, 281 s. ISBN 80-01-01071-6.
[83] VLČEK, K. *Teorie informace, kódování a kryptografie*. 1. vyd. Ostrava: VŠB–TU Ostrava, 1999, 184 s. ISBN 80-7078-614-0.
[84] VONDRUŠKA, P. *Kryptografie, šifrování a tajná písma*. Praha: Albatros, 2006, 342 s. ISBN 80-00-01888-8.
[85] VŠETIČKA, V. Co by ani Sir A. C. Doyle nevyřešil. *Bajt*, 1992, č. 3/92, s. 125-126.
[86] ZELINKA, I., OPLATKOVÁ, Z., ŠEDA, M., OŠMERA, P. & VČELAŘ, F. *Evoluční výpočetní techniky. Principy a aplikace*. 1. vyd. Praha, BEN – Technická literatura, 2009. ISBN 978-80-7300-218-3.