Prezentačná vrstva

Ciele

Čo by mal študent vedieť:

- ✓ dôvody pre špecifikovanie prezentačnej vrstvy
- ✓ funkcie prezentačnej vrstvy (konverzia dát, kódovanie zdroja, kompresia, šifrovanie)
- ✓ princípy používané na prezentačnej vrstve
- ✓ protokoly prezentačnej vrstvy

Úvod

Aplikačná vrstva definuje spôsob, akým komunikujú so sieťou aplikácie. Ostatné vrstvy sa starajú o to, aby prenášané dáta prišli ku koncovému príjemcovi presne také, ako boli vyslané. Avšak rovnaká podoba dát ešte neznamená, že pre príjemcu budú prenesené dáta predstavovať to iste, ako pre ich odosielateľa. Interpretácia dát môže byť rôzna napríklad z nasledovných dôvodov:

- Niektoré počítače firmy IBM používajú pre kódovanie znakov kód EBCDIC, zatiaľ čo prevažná väčšina ostatných používa k rovnakému účelu kód ASCII.
- K znázorneniu celých čísiel so znamienkom používa väčšina počítačov tzv. dvojkový doplnkový kód, ale napríklad. počítače CDC Cyber pracujú s jednotkovým doplnkovým kódom.
- Mikroprocesory 80x86 firmy Intel číslujú jednotlivé byty jedným smerom, zatiaľ čo mikroprocesory radu M68000 firmy Motorola číslujú jednotlivé byty presne opačne.
- Veľmi časté sú odlišnosti napríklad vo formáte čísiel v pohyblivé rádovej čiarke, odlišné rozsahy zobraziteľných celých čísel, dané počtom k tomu vyhradených bitov a podobne.

Všeobecne možno povedať, že rôzne počítače používajú rôzne spôsoby vnútornej reprezentácie dát. Ak si majú takéto počítače korektne odovzdávať dáta, musí byť zaistený buď rovnaký formát dát alebo vykonaná potrebná konverzia jedného formátu dát do iného formátu. A to je v referenčnom OSI modeli úloha **prezentačnej vrstvy** (**presentation layer**). Prezentačná vrstva správne určí typ a formát vyslaných/prijatých dát, a v prípade potreby vykoná potrebnú konverziu.

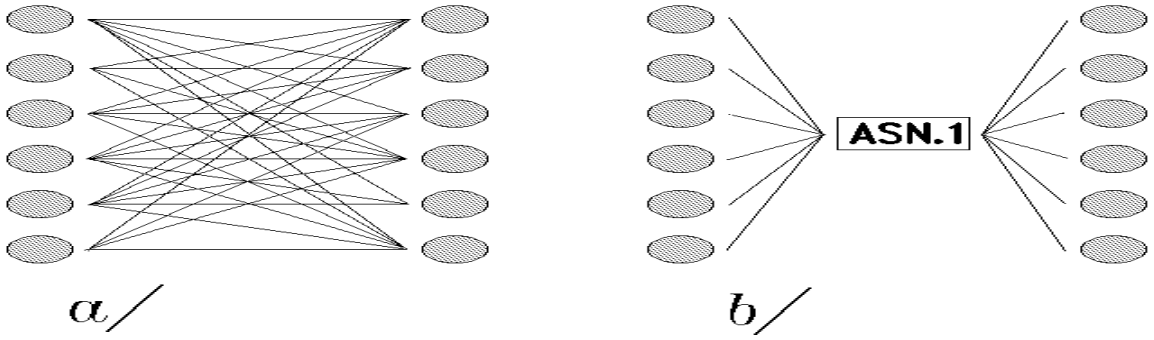
Prezentačná vrstva sa stará o to, aby napríklad celé číslo bez znamienka s hodnotou 234 bolo prenesené ako celé číslo bez znamienka s hodnotou 234, a nie ako celé číslo so znamienkom s hodnotou -22. Nezaujíma sa, čo to číslo vyjadruje., či je to finančná suma alebo číslo domu To je záležitosť aplikácie alebo jej používateľa.

Na úrovni prezentačnej vrstvy je ešte riešený problém minimalizácie objemu prenášaných dát, ktorý je označovaný ako **komprimácia** (**compression)**.

S prezentačnou vrstvou je spojené aj riešenie zabezpečenia prenášaných dát pomocou **šifrovania** (**encryption**).

Konverzia dát

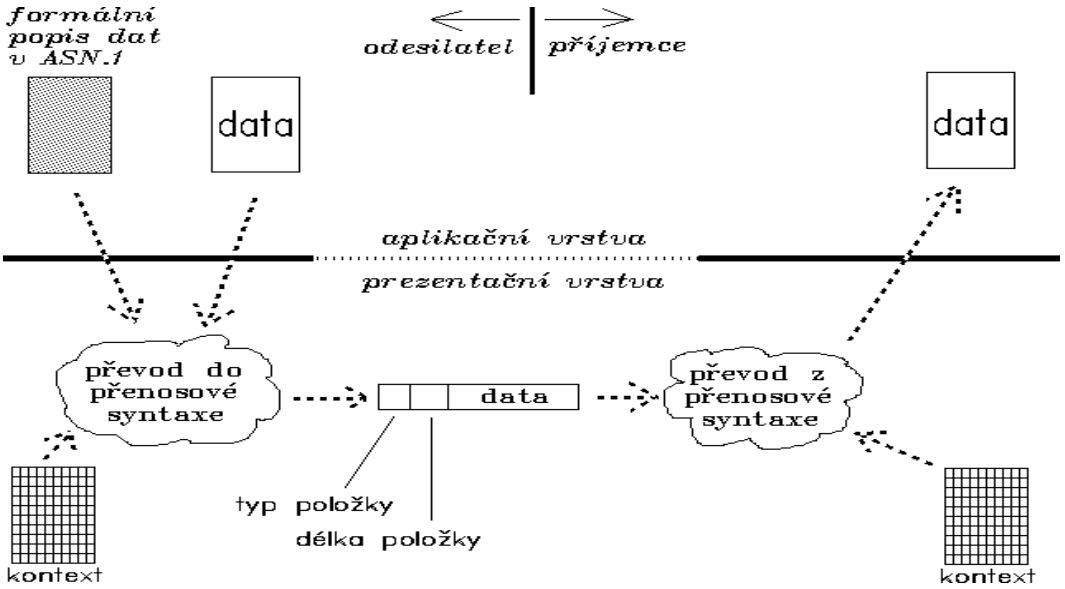
Pre zaistenie konverzií na úrovni prezentační vrstvy sa ponúkajú dve základné možnosti podľa obr. 1. Prvá z nich predstavuje vzájomné priame prispôsobenie štýlu "každý s každým", při kterém jsou přenášená data konvertována jen jednou - jsou-li ovšem k dispozici nezbytné konverzní rutiny pro libovolnou dvojici komunikujících uzlů. Ve druhém případě jsou přenášená data konvertována dvakrát: ze tvaru, se kterým pracuje odesilatel, jsou nejprve převedena do společného "mezitvaru", a z něj pak do takového tvaru, s jakým pracuje jejich příjemce. Nevýhoda dvojí konverze je zde kompenzována tím, že pro každou konkrétní reprezentaci dat, se kterou pracuje nějaký počítač, stačí jediná konverzní utilita pro jejich převod z/do společného "mezitvaru".



Obr.1.: Možnosti konverzie
a/ typu "každý s každým"
b/ so spoločným "mezitvarom"

Referenční model ISO/OSI předpokládá právě tuto druhou variantu se společným mezitvarem. Podívejme se proto na její podstatu poněkud podrobněji.

Chtějí-li vzájemně spolupracovat dvě různé síťové aplikace, musí se nejprve domluvit na společných datových strukturách, které budou používat - tedy například na tom, že datum budou reprezentovat jako záznam (record) tvořený třemi položkami (DEN, MESIC a ROK), které jsou samy o sobě celými čísly bez znaménka. Tyto datové struktury je ovšem nutné vyjádřit tak, aby jejich popis byl pro obě strany srozumitelný, a obě strany si jej také stejně vykládaly. Kdyby byly všechny síťové aplikace psány v jediném vyšším programovacím jazyku, stačilo by použít právě tento jazyk. Předpoklad použití jediného programovacího jazyka však nebyl, není a zřejmě nikdy nebude v praxi splněn, a tak bylo nutné vytvořit pro potřeby formálního popisu dat a datových struktur zvláštní jazyk, který byl nazván **ASN.1** (**Abstract Syntax Notation**). Umožňuje definovat jednotlivé datové položky, stanovit jejich typ (tj. určit, zda jde např. o celé číslo se znaménkem, znakový řetězec či logickou hodnotu apod.), přidělit jim jméno (identifikátor), a také sestavit z jednoduchých datových položek obecnější datové struktury typu záznam, pole, seznam, množina apod.



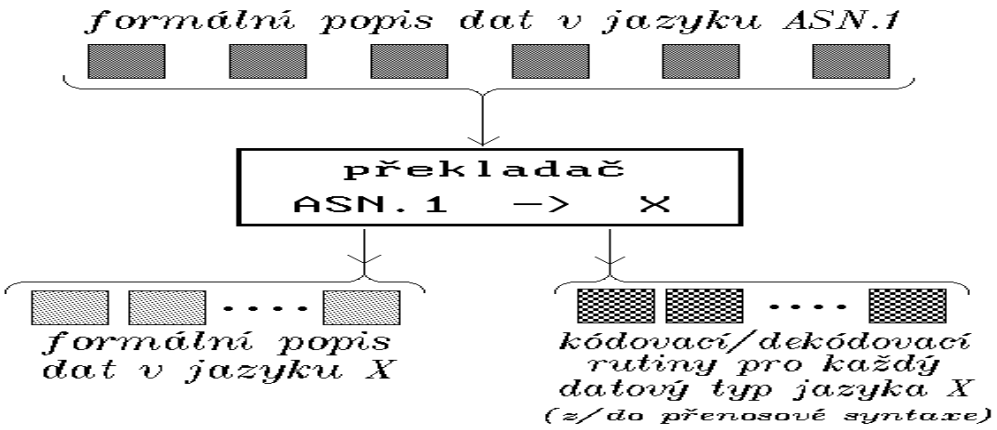
Obr. 2. Predstava prezentačnej vrstvy

Jazyk ASN.1, který vzdáleně připomíná jazyk Pascal, je tedy jazykem pro formální popis dat - což se v terminologii ISO/OSI modelu označuje jako **abstraktní syntaxe** (**abstract syntax**). Abstraktní proto, že ještě neurčuje žádný konkrétní způsob reprezentace těchto dat. Pro potřeby vlastního přenosu je ale samozřejmě nutné veškerá data vhodným způsobem zakódovat. Způsob kódování datových struktur jazyka ASN.1 pro potřeby jejich přenosu pak určuje samostatná norma organizace ISO (IS 8825). Formát skutečně přenášených dat se přitom v terminologii ISO/OSI modelu označuje jako **přenosová syntaxe** (**transfer syntax**). Její konkrétní tvar je založen na myšlence, že každá jednotlivá položka dat by měla být samoidentifikující, tedy měla by sebou nést i informaci o vlastním typu. Každá datová položka proto má při přenosu tři části, které po řadě určují její typ, délku a vlastní obsah (viz obrázek 2.).

Způsob fungování prezentační vrstvy názorně ilustruje obrázek 2. Kdykoli chce nějaká entita aplikační vrstvy zaslat určitá data své partnerské entitě na jiném uzlovém počítači, předá "své" prezentační vrstvě jednak vlastní data, která si přeje odeslat, a jednak jejich popis v jazyce ASN.1 (který definuje jejich abstraktní syntaxi). Prezentační vrstva na základě tohoto popisu dokáže správně interpretovat jednotlivé položky dat (určit mj. jejich typ a velikost), a na základě toho je pak zakódovat do takového tvaru, který je vhodný pro přenos, a který si sebou nese potřebné informace o typu a formátu přenášených dat (tj. převést je do přenosové syntaxe). Prezentační vrstva na straně příjemce pak díky tomu dokáže správně určit typ a formát přijatých dat, a v případě potřeby provést nezbytné konverze. Jestliže například přenosová syntaxe počítá s vyjádřením celých čísel se znaménkem ve dvojkovém doplňku, ale příjemce používá ke stejnému účelu jednotkový doplněk, může prezentační vrstva příjemce provést nezbytné konverze ještě dříve, než přijatá data předá své bezprostředně vyšší (tj. aplikační vrstvě).

Prezentační vrstvy příjemce a odesilatele se však nejprve musí shodnout na tom, jaké datové struktury si vlastně budou předávat, a jakou budou pro ně používat přenosovou syntaxi. Proto se musí obě strany na začátku vzájemného spojení (přesněji: při zahajování relace) nejprve dohodnout na jednom nebo několika tzv. **kontextech,** jak se v terminologii ISO/OSI modelu

nazývá přiřazení přenosové syntaxe k syntaxi abstraktní. V průběhu relace se pak mohou mezi těmi kontexty, na kterých se oba dohodli, dokonce přepínat.



Obr. .3.: Představa překladače jazyka ASN.1

V současné době je praktické používání jazyka ASN.1 značně usnadněno existencí překladačů z tohoto jazyka do obvyklých vyšších programovacích jazyků, např. do Pascalu či jazyka C. Jak názorně ukazuje obrázek 3., je vstupem tohoto překladače popis datových struktur v jazyku ASN.1, a výstupem jednak ekvivalentní popis v příslušném cílovém jazyku (tedy např. v C či v Pascalu), který pak lze přilinkovat ke zdrojovému tvaru vlastní aplikace, a dále také kódovací a dekódovací rutiny (určené pro potřeby prezentační vrstvy), které převádí datové struktury z příslušného cílového jazyka přímo do jejich přenosové syntaxe resp. obráceně.

V architektúre TCP/IP sa používajú iné štandardy:

Multipurpose Internet Mail Extensions (MIME) je internetový štandard, ktorý rozširuje základný formát e-mailu o podporu:

- používať text, ktorý nie je v znakovej sade ASCII (napr. písať správu s diakritikou)
- používať netextové prílohy
- používať (skladať) telo emailu z viacerých častí
- používať v hlavičkách emailu informácie, ktoré nie sú v znakovej sade ASCII (napr. použiť meno odosielateľa s diakritikou)

MIME je taktiež používaný nielen na popis obsahu emailu, ale aj obsahových typov (*content types*) vo všeobecnosti.

Obsahové typy definované v MIME štandarde sú potrebné aj mimo emailov, napríklad pri komunikačných protokoloch ako HTTP služby WWW. Pretože HTTP vyžaduje prenos dát vo formátovaní podobnom emailu.

Základný prenosový protokol pre email - SMTP, podporuje len 7bitové ASCII znaky. Toto limitovalo použitie emailu v Internete na správy písané v angličtine. Mnohé ostatné jazyky založené na latinske využívajú diakritiku, ktorá v ASCII kódovaní nemá vyhradené znaky.

Práve MIME definuje mechanizmus ako pred prenosom zakódovať správy v iných jazykoch, dokonca nielen v latinke. MIME umožňuje zasielať aj 8bitové binárne dáta ako sú obrázky, zvuky, videá, či spustiteľné programy. Kódovanie správ do a spätne z MIME formátu automaticky prevádza e-mailový klient alebo podprogram bežiaci na emailovom serveri (spoločný názov oboch je MUA (*Mail User Agent*)).

Základný formát emailu je daný RFC 822, aktualizovaný v RFC 2822. Tieto štandardy používajú kľučové (vyhradené) slová, ktoré sú uvedené v hlavičkách emailu. Pokiaľ za kľúčovým slovom nasleduje hodnota (je to kompletná hlavička) MUA alebo SMTP server ju môže jednoznačne interpretovať. Typicky sú to základné hlavičky `From:, To:, Subject:, Date:`.

MIME zavádza nové hlavičky, ktoré umožňujňujú špecifikovať obsah (*content-type*) alebo kódovanie (*transfer encoding*) jednotlivých častí v tele emailu. Teda je možné MUA informovať dopredu o type príloh emailu, ktorý následne môže zobraziť ikonu, zobraziť rôzne ponuky aj bez analýzy obsahu týchto príloh. Použité prenosové kódovania zasa spĺňajú obmedzenie dané SMTP servermi - prenos len 7bitových ASCII znakov. Ďalej je možné uviesť v samotnej základnej hlavičke text s diakritikou (príslušne zakódovaný do ASCII).

MIME je rozšíriteľné. To znamená, že štandard umožňuje použiť aj iné, nové obsahové typy a nové vlastnosti MIME objektov (jednotlivých častí emailu), ktoré ale nemusia všetky MUA správne interpretovať. Cieľom zavedenia takéhoto zavedenia rozšírenia emailu - MIME, bolo nijak nepozmeniť štandardy pre základný formát emailu. Preto aj MUA, ktoré nepodporujú MIME, dokážu správu spracovať a aspoň čiastočne zobraziť.

- ☐ **Type-Length-Value (TLV)**
- ☐ **Basic Encoding Rules**

Kódovanie všeobecne

Kódovanie je proces, pri ktorom sa každému znaku alebo postupnosti znakov daného súboru znakov (vzorov) jednoznačne priradí znak alebo postupnosť znakov (obrazov) z iného súboru znakov.

Kódovanie **je teda transformácia** určitej **informácie z jednej formy na druhú pomocou** určitého postupu - **algoritmu**, ktorý je väčšinou verejne známy. Vo väčšine prípadov teda účelom kódovania nie je utajenie informácie (na rozdiel od šifrovania) ale len jej iná forma zápisu vybrané tak, aby sa informácia dala čo najlepšie alebo najúspornejšie uchovať alebo preniesť.

Vďaka počítačom sa najčastejšie sa používa kódovanie údajov a informácií do číselnej podoby. Takémuto kódovaniu tiež hovoríme Digitalizácia. Používa sa však aj nečíselné kódovanie.

Najčastejším problémom, pre ktorý sa neustále hľadajú nové riešenia **je nájdenie nových úspornejších spôsobov kódovania pre čísla, znaky, text, zvuk, grafiku a video**. Najčastejšie spôsoby kódovania jednotlivých údajov, pre ktoré sú v počítači definované určité operácie (sčítanie, násobenie, spájanie…), sú v počítači reprezentované dátovými typmi.

Kódovanie čísel v počítači

Všetky údaje v počítači sú kódované pomocou rôznej kombinácie hodnôt bitov [b] - najmenšej jednotky informácie. Každý z bitov môže nadobúdať iba dve rôzne hodnoty **0 a 1.** Tieto **bity sú však do pamäťových buniek počítača ukladané po osem, preto je výhodné na zakódovanie údajov použiť vždy taký počet bitov, ktorý je deliteľný ôsmimi (8,16,24,32 ....).** Čím väčší počet bitov použijeme, tým väčší rozsah čísel môžeme použiť. Napríklad pomocou 8 bitov dostaneme $\mathbf{2^8 = 256}$ rôznych **kombinácií** núl a jednotiek. **Pomocou 8 bito**v teda môžeme zakódovať napríklad čísla **od 0 do 255** alebo čísla **od -128 do 127** v prípade, že potrebujeme i záporné čísla. ***Na kódovanie čísiel v počítačoch je najvýhodnejšie použiť jedno „slovo" (Word), t.j. taký počet bitov, ktoré počítač dokáže spracovať počas jednej operácie (jedného taktu procesora).*** Najmodernejšie počítače dnes používajú 64 bitové slovo, teda dokážu spracovať 64 bitov pri jednej operácií.

Kódovanie prirodzených čísel a nuly

Obrovskou výhodou je fakt, že každé číslo môžeme previesť do dvojkovej sústavy, ktorá používa iba cifry 0 a 1, čím pre každé číslo dostaneme jednoznačný zápis. Prirodzené čísla sú teda v počítači uložené v tzv. priamom kóde, čo je vlastne číslo prevedené do dvojkovej sústavy.

Pri použití jedného Bajtu (8 bitov) môžeme zakódovať 256 možných hodnôt, t.j. hodnoty 0 až 255 Pri použití 2 Bajtov (16 bitov) hodnoty 0 až 65535 Pri použití 4 Bajtov (32 bitov) hodnoty 0 až 4 294 967 295 Pri použití 1 slova moderného počítača (64 bitov) hodnoty 0 až

18 446 744 073 709 551 615.

V niektorých prípadoch (napríklad pri prenose) je vhodnejšie použiť iný kód ako je zápis čísla v dvojkovej sústave. Jedným z takýchto kódov je kód BCD.

Kód BCD (Binary Coded Decimal) je jedným z najčastejšie používaných kódov na reprezentovanie desiatkových čísel. **Pri tomto kóde je každá desiatková číslica zakódovaná pomocou štyroch bitov.**

Tento kód kvôli rôznym výhodám či nevýhodám **bol rôzne modifikovaný**. Boli pozmenené váhové stavy jednotlivých bitov kódu, preto sa začali tieto váhy dopisovať za označenie kódu BCD. ***Klasickému kódu prislúcha kód 8421, kde prvý bit s prava má váhu čísla 1, druhý váhu čísla 2, tretí váhu čísla 4 a posledný váhu čísla 8***. Napríklad číslo BCD 8421 kóde 0111 predstavuje číslicu 7 (8.0 + 4.1 + 2.1 + 1.1). Známe sú tiež kódy ***BCD 2421, BCD 84-2-1*** alebo kód ***BCD 8421*** firmy IBM, ktorý nulu kódoval bitmi 1010.

Kódovanie celých čísel

Pri celých číslach je potrebné zohľadniť i znamienko. ***Pretože sú znamienka len dve (+, -), môžeme ich zakódovať pomocou jedného bitu (0 = +, 1= -).*** Pri kódovaní celých čísel sa znamienko zakóduje vždy prvým bitom zľava. Napr. pri použití 1 Bajtu bude 10011101 kód pre -29.

-Pri použití 1 Bajtu (8 bitov - 1 bit znamienko a 7 bitov hodnota), môžeme zakódovať hodnoty -128 až +127

-Pri použití 2 Bajtov (16 bitov -1 bit znamienko a 15 bitov hodnota), môžeme zakódovať hodnoty -32 768 až +32 767

-Pri použití 4 Bajtov (32 bitov - 1 bit znamienko a 31 bitov hodnota), môžeme zakódovať hodnoty -2 147 483 648 až +2 147 483 647

-Pri použití 4 Bajtov (64 bitov - 1 bit znamienko a 63 bitov hodnota), môžeme zakódovať hodnoty -9 223 372 036 854 775 808 až +9 223 372 036 854 775 807

Kódovanie reálnych čísel

Reálne čísla môžeme do počítača kódovať dvoma spôsobmi:

Ako **čísla s pevnou rádovou čiarkou** (tento spôsob sa väčšinou používa na uloženie meny napr.: 24,50 Sk). Pri tomto spôsobe je **niekoľko bitov vyhradených pre celú časť čísla** a **niekoľko pre desatinnú časť čísla**. Ak nám pri nejakej operácii dostaneme väčší počet desatinných miest ako môžeme zakódovať pomocou vyhradeného počtu bitov, vtedy sa **zvyšné miesta jednoducho odrežú a nebudú do pamäte počítača uložené.**

Ako **čísla s pohyblivou rádovou čiarkou** – tu je vyhradených **niekoľko bitov pre hodnotu čísla (mantisu) a zvyšok je vyhradený pre exponent**. Napr.: číslo 126,567 je uložené ako 126567.10-3. V našom prípade je mantisa 126567 a exponent -3. Obe tieto hodnoty sú uložené samozrejme v priamom kóde a majú 1 bit vyhradený pre znamienko.

Kódovanie znakov

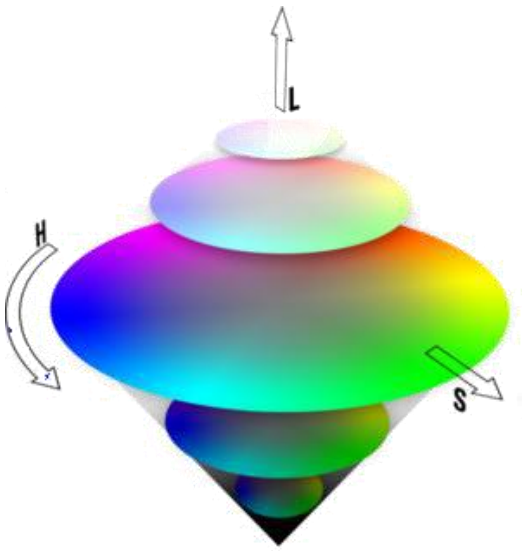
Na rozdiel od čísel, **znaky textu nevieme previesť do dvojkovej sústavy**, preto bolo potrebné vymyslieť iný spôsob ako jednoznačne priradiť určitému znaku práve jednu kombináciu núl a jednotiek, ktorá tento znak v počítači bude reprezentovať. **Keďže neexistuje žiadny univerzálny spôsob ako to urobiť, každý výrobca počítačov tento problém riešil iným spôsobom, preto existuje viacero znakových kódov.** Poriadok do tohto chaosu sa snažil zaviesť americký úrad pre normalizáciu, ktorý vyhlásil jeden spôsob, ktorý by mali všetci používať. Tento spôsob kódovania sa volá **ASCII - American Standard Code for Information Interchange** (Americký štandardný kód pre výmenu informácií).

Tento štandard hovorí, že **na zakódovanie každého znaku sa použije 8 bitov. Čo umožňuje definovať kód pre 256 znakov**. Pričom **prvá polovica znakov bude pre všetky krajiny rovnaká** a zvyšných **128 znakov sa pre každú krajinu** stanovil **podľa ich potrieb.** Tento spôsob vniesol do kódovania znakov neuveriteľný chaos, preto sa vymyslel nový spôsob kódovania UNICODE.

UNICODE - je **medzinárodný štandard, ktorého cieľom je definovať jedinečný kód zvaný code point** pre každú z grafém používaných na zápis ľudského jazyka**. Vznik jednotného kódu znakov, podmienila existencia množstva znakových sád**. Znakové sady sa líšili nielen pre jednotlivé krajiny, ale i v rámci jednej z krajín existovalo viacero znakových sád. Kódovanie UNICODE je navrhnuté pomocou dvojbajtového kódovania znakov, čo **umožňuje vytvárať sady s 65 536 znakmi**. Štandard je navrhnutý tak, že všetky možné znaky rozdeľuje do sedemnástich dvojbajtových plánov. Takéto rozdelenie umožňuje definovať až 1 114 112 (= 17 × 216) znakov. Prvá verzia štandardu vznikla v októbri 1991 no **súčasná verzia štandardu je už UNICODE 5.0**. Táto definuje 101 063 znakov čo je iba 9,1 % zo všetkých možných.

Kódovanie farieb

Aby bol obrázok farebný, musíme každému bodu - pixelu obrázka priradiť určitú farbu. Spôsobov, akými sa to dá urobiť je opäť niekoľko. Na to, aby sme to vedeli urobiť, musíme vedieť niečo o ľudskom oku a spôsobe videnia farieb. ***Farby vnímame vďaka svetlu, ktoré vstupuje do nášho oka a dopadá na bunky - čapíky***, ktoré sú citlivé na farbu. Svetlo je vlastne elektromagnetické vlnenie s rôznou vlnovou dĺžkou. ***Ľudské oko dokáže vnímať iba určitý rozsah vlnových dĺžok svetla***. Svetlo s najväčšou vlnovou dĺžkou, ktoré ľudské oko vníma, je červenej farby (780 nm) a svetlo s najnižšou vlnovou dĺžkou je svetlo fialovej farby (380nm). V tomto spektre farieb je **ľudské oko schopné** bez problémov **rozlíšiť vyše 1,5 milióna rôznych farieb.** Nie však každej farbe, ktorú rozlíšime pripadá iná vlnová dĺžka, pretože podľa intenzity dopadajúceho svetla, ľudské oko rozlišuje sýtosť a jas daného odtieňa farby. Takže iná vlnová dĺžka prislúcha iba rôznym odtieňom farieb.

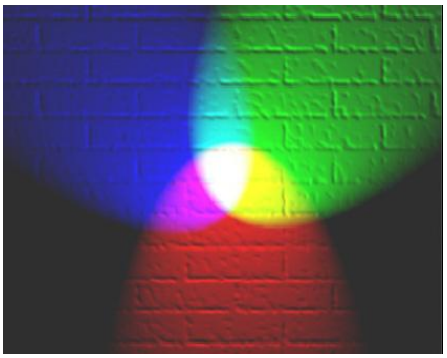


Farebné modely HSB a HLS

Prvé riešenie problému kódovania farieb, ktoré sa ponúka, je zobrať **celý rozsah vlnových dĺžok** a **rozdeliť** ho **na niekoľko častí a do pamäte počítača uložiť poradové číslo farby, ktorej prislúcha určitá vlnová dĺžka**, ďalej jej **sýtosť a jas**. Kódovanie farieb takýmto spôsobom využívajú farebné modely HSB (či HSV), HLS **Oba modely uchovávajú informáciu o. odtieni (Hue)** a **sýtosti (Saturation**). Odlišujú sa iba v tom, že **prvý model**

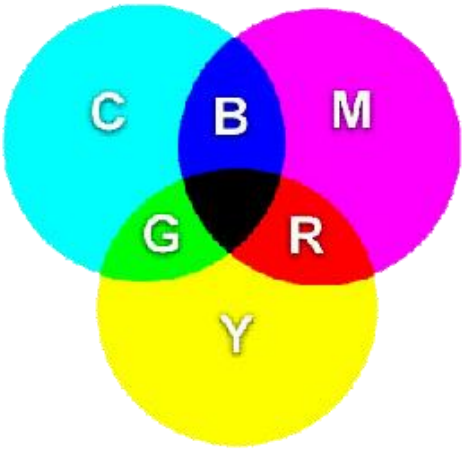
používa jas (Brightness) a **druhý používa svetlosť (Lightness).** V prvom modeli dostaneme čiernu farbu tým, že nastavíme jas na nulu a bielu tak, že jas nastavíme na maximálnu hodnotu (nezávisle od sýtosti). V druhom modeli dosiahneme čiernu, ak je svetlosť aj sýtosť 0 a bielu, ak je svetlosť aj sýtosť maximálna. **Hlavným nedostatkom tohto problému je náročnosť vyrobenia svetla so zadanou vlnovou dĺžkou**. Naštastie existujú aj iné riešenia problému kódovania farieb.

Farebný model RGB



Farebný model **RGB je najčastejšie využívané kódovanie farby bodu obrázka**. Empiricky sa zistilo, že **takmer všetky farby sa dajú vytvoriť zmiešaním ľubovoľných troch nezávislých farieb** (t.j. že pomocou zmiešania dvoch nedostaneme tretiu). Najvýhodnejšie pre výrobu svetelných lúčov (aditívny farebný systém) bolo použitie farieb **červená *(Red)*, zelená *(Green)*, modrá *(Blue)***. Aby sme vedeli vytvoriť 1,5 milióna farieb stačí, ak **každú z týchto farieb rozdelíme na 115 oddtieňov**, ktorých zmiešaním v rôznych pomeroch vzniknú všetky farby. Kvôli uchovaniu v pamäti počítača, je však výhodnejšie použiť až **256 odtieňov každej farby (8 bitov)**, čo **nám umožní vytvoriť až 16 777 216 rôznych farieb.** Pri takomto kódovaní je každá farba zakódovaná 24 bitmi, čo sú tri pamäťové miesta počítača. Pričom **255 0 0 je sýta červená** farba, **0 255 0 je sýta zelená** farba, **0 0 255 je sýta modrá farba**, **0 0 0 je čierna** farba a **255 255 255 je biela farba.**

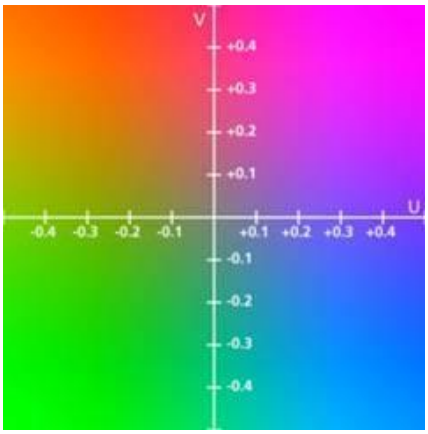
Farebný model CMY



Farebný model **CMY je model, ktorý používa doplnkové farby** (substraktívny farebný systém). V modeli RGB platilo, že ak zmiešame všetky tri základné farby s maximálnou sýtosťou, dostaneme bielu farbu. Takýto spôsob je výhodný pri obrazovkách monitorov, pretože tienidlo je čierne. **Pri tlačiarňach sa** však **tlačí na biely papier, preto potrebujeme** vziať **také farby**, pri ktorých, **ak zmiešame ich najsýtejšie odtiene, dostaneme čiernu farbu.** Takéto farby dostaneme, keď zoberieme doplnkové farby k farbám červená, zelená a modrá. Týmito farbami sú **azúrová (Cyan)**, **purpurová (Magenta)** a **žltá (Yellow**). Kvôli tomu, že je lacnejšie vyrobiť čierny atrament ako ho miešať pomocou týchto troch farieb, sa k týmto farbám pridáva i samostatná čierna farba a tento model sa označuje tiež CMYK, kde posledné písmeno je odvodené od blacK - čierna . Výhodou tohto formátu je tá, že **výsledná farba CMY sa dá veľmi jednoducho získať z modelu RGB pomocou vzorcov:**

C = (255 - R); M = (255 - G); Y = (255 - B)

Farebný model YUV



Farebný model **YUV slúži na zachovanie čiernobielej informácie pri televíznom vysielaní**. Po vzniku farebného filmu nastal problém, ako zakódovať farbu tak, aby farebné filmy mohli pozerať i ľudia s čiernobielymi prijímačmi. Bolo potrebné zachovať pôvodnú čiernobielu informáciu a doplniť ju tak, aby vznikol farebný obraz. Farba je kódovaná tak, že **k čiernobielej zložke Y, ktorú tiež nazývame svietivosť (luminance), pridáme dve farebné zložky UV farebnosti (chrominance)**, pričom zložka **U udáva odtieň medzi modrou a žltou** a zložka **V udáva odtieň medzi červenou a žltou** farbou. Takže čiernobiely prijímač berie do úvahy iba zložku Y, farebné prijímače za pomoci zložiek YUV získajú **RGB kód farby pomocou jednoduchej transformácie:**

R = Y + 1,403V; G = Y - 0,344U - 0,714V; B = Y + 1,770U

Komprimácia

Komprimácia (kompresia, pakovanie či balenie) dát **je proces, pri ktorom sa znižuje objem dát, pričom existujú dva druhy komprimácie:**

- **nestratová** - pri ktorej **nedochádza k strate údajov**. To znamená, že ak skomprimované dáta dekomprimujeme, získame úplne identické dáta. Takto sa balia napríklad textové, programové a iné súbory. **Kompresia sa deje na základe vynechania redundantných** (nadpočetných) **informácií. Kompresný pomer**, ktorý predstavuje pomer medzi veľkosťou dát pred spakovaním a po ňom, sa tu dá dosiahnuť až **okolo 2:1**, niekedy aj viac, to však závisí od druhu dát.

- **stratová** - je proces, pri ktorom sa **vynechajú tie údaje, ktoré sú pre celkový dojem z dát nepodstatné. Kompresný pomer je niekedy až 200:1**, ale dáta sa už po kompresii nikdy nedajú zrekonštruovať do pôvodnej podoby. Časť informácií totiž chýba. **Stratovú koprimácia používa hlavne pre komprimovanie mediálnych súborov a to zvuk, obraz, video ...** Aj keď komprimačný pomer sa zdá veľmi veľký, aj pri niekoľko desaťnásobnej redukcii dát je výsledok komprimácie takmer nerozoznateľný od originálu. Platí to najmä pre obrázky vo formáte JPG, alebo video vo formáte MPEG. Cenou za to je veľmi zložitý postup komprimácie a nutnosť použiť zložité matematické metódy.
  Z princípu stratovej komprimácie jednoznačne vyplýva, že nie je možné komprimovať touto metódou napríklad textový súbor. Po jeho dekompresii by sme nedostali čitateľné dáta. Obrázok by sa, naopak, dal skomprimovať pomocou nestratovej komprimácie, pričom by sme dosiahli komprimačný pomer obvyklý pre nestratovú komprimáciu. Táto metóda je výhodná vtedy, ak chceme znížiť ojem dát aspoň v menšom komprimačnom pomere, ale sa nechceme vzdať detailov, ktoré by sa pri stratovej komprimácii stratili.

Komprimácia sa používa najčastejšie na zníženie objemu multimediálnych dát pri prenose dát v sieťach (v internete).

Komprimačné formáty:

- Audio: mp3,ogg, mp2, wma, ...
- Video: mpeg, xvid, wmv, divx, ...
- Obrázky: jpeg, gif, ...
- Dáta: zip, rar, arj, cab, tar, gz,

Komprimácia sa vykonáva buď automaticky (uložením súboru v komprimovanom formáte JPG, MPEG, MP3) **alebo pomocou špeciálneho komprimačného programu** (ZIP, RAR). **Dekomprimácia sa deje buď samo rozbalením alebo pomocou špeciálneho dekomprimačného programu. Niektoré dáta ostávajú trvale vo svojej komprimovanej podobe** (najmä pri stratovej kompresii).

Výhodou archivačných formátov ako sú WinRar, WinZip, 7Zip je, že dokážu do jedného balíka vložiť množstvo súborov čo je výhoda, ktorú určite oceníme napríklad pri mailovaní. **Ak však archivujeme súbory len za účelom zníženia ich veľkosti je dobre si uvedomiť, že komprimovať sa oplatí len súbory, na ktorých ešte nie je aplikovaná žiadna kompresia.** Aj preto majú napríklad obrázky vo formáte TIFF alebo BMP po pridaní do archívu veľmi malú veľkosť, zatiaľ čo obrázky v JPEG nezmenšíme takmer vôbec. To isté platí aj o hudobných súboroch. **Veľmi efektívne je komprimovať textové dokumenty, databázy, súbory s poštou.**

Čo sa komprimuje?

Komprimujú sa dáta – napríklad fotka, akú nafotí fotoaparát je dátovo veľmi veľká.

Digitálna fotka je počítačový súbor, ktorý prenáša tri informácie o každom jednom obrazovom bode, z ktorého sa skladá obraz. Ako príklad použime obrázok z trojmegapixelového prístroja, napríklad v rozmere 2048 × 1536 pixelov. Taký obrázok sa skladá z 3145728 obrazových bodov. Tieto tri milióny treba násobiť tromi – každý bod totiž musí obsahovať farebnú informáciu a tá sa skladá z troch rôznych farieb. Červenej, modrej a zelenej. Farebný obraz v rozmere 2048 × 1536 pixelov sa v skutočnosti skladá z troch čiernobielych fotiek rovnakých rozmerov. Preto je skutočná veľkosť farebnej fotky výsledkom násobenia výška krát šírka obrázku a to celé krát tri.

Tri čiernobiele obrázky tak spolu vytvoria úctyhodných 9437184 obrazových bodov, ktoré sú nutné na to, aby sa dala preniesť informácia, ktorú označujeme, ako „trojmegapixelový farebný obrázok“. Napríklad 5 megapixelový záber s rozmermi 2608 × 1952 pixelov má dátový objem 15 mega.

Upozorňujem, že uvedené sa týka hotovej digitálnej fotky, nie je to popis, ako pracuje snímač – tam je situácia iná.

Takých 9 a pol mega je však dátovo náročný objem. Už len jeho uloženie na pamäťovú kartu dnes v bežných aparátoch trvá asi 25 sekúnd. Navyše pamäťové karty majú menšiu kapacitu – na jednu 128 megovú kartu by sa vošlo asi 12 obrázkov. Také fotenie by naskutku nikoho nebavilo.

Tieto dáta preto treba nejako upraviť, aby boli menšie. A tu prichádza k slovu kompresia.Dáta sa stlačia tak, aby obrázok zaberal menší dátový priestor, napríklad desatinu pôvodného a pritom, aby sa zachovala podľa možnosti plnohodnotná obrazová informácia. Komprimuje sa teda veľkosť dát súboru s fotkou.

Ako sa dáta komprimujú?

Popisovaný princíp kompresie je len príkladom.

Obrazové dáta sú v digitálnom fotoaparáte zaznamenané, ako veľmi dlhý rad čísiel.

Fotoaparát, aj počítač „vedia“, že číselné údaje reprezentujú obrazové body. Je dohodnuté, v akom poradí sa informácie o bodoch obrázka ukladajú. Na začiatku každého súboru je akýsi „predpis“, ako s dátami naložiť. V počítačovej reči tam je povedané niečo ako „nakresli obrázok, ktorý bude mať šírku 2048 bodov a výšku 1536 riadkov. Posielam dáta po horizontálnych riadkoch, umiestni ich zľava doprava“. A ďalej je už len dlhý tok čísiel, o ktorých už ale je známe kam patria.

Fotka často krát zaberá veľmi podobné oblasti. Napríklad farba oblohy je modrá, alebo je na obrázku nejaká biela stena.

Ak sú dáta podobné, alebo dokonca rovnaké dajú sa stlačiť.

Predstavme si, že malý kúsok dátového súboru vyzerá takto:

31 31 31 31 36 36 36 36 36 36 36 36 36 36 38 38 38

(Čísla reprezentujú stupne šedej v rozsahu 0 – 255 pre jednu farbu, inými slovami pre jednu čiernobielu fotku.)

Dajú sa stlačiť tak, že aparát nezapíše opakujúce sa hodnoty. Zapíše iba to, koľkokrát sa opakujú. Napríklad:

4×31 10×36 4×38

A hneď tu máme úsporu - počet čísel, potrebných na prenesenie informácie o riadku čísel poklesol na polovicu. Pri prezeraní fotky bude síce potrebný najskôr výpočet pôvodných dát, ale to spravidla nevadí. V súčasnosti je výkon výpočtových procesorov lacnejší a rýchlejší, ako prístup k úložnému priestoru.

V skutočnosti je kompresia fotiek oveľa zložitejšia – berie sa do úvahy aj porovnanie kanálov a mnoho iných parametrov o ktorých nemusíte nič vedieť.

Stačí, ak vieme, že kompresiou sa podarí zmenšiť objem dát, ktoré sa musia preniesť z fotoaparátu do počítača. V počítači sa potom obrázok roztiahne na pôvodnú dátovú veľkosť vždy, keď si ho chcete prezrieť – to za nás automaticky spraví prehliadač fotiek.

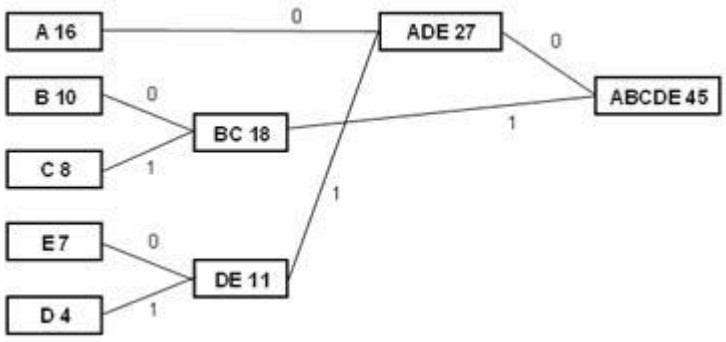
Ako je realizovaná kompresia?

Huffmanovo kódovanie

Huffmanovo kódovanie patrí medzi najrozšírenejšie bezstratové statické kompresné metódy. Tvorcom tohto kódovania je David A. Huffman, ktorý ho vymyslel ešte v päťdesiatych rokoch 20. storočia a odvtedy našiel využitie pri kódovaní JPEG (JPEG je síce stratová kompresia, ale pozostáva aj z tejto bezstratovej metódy keďže pri kompresii sa využíva niekoľko komprimačných metód), MP3, ZIP formátoch, ale aj videa.

Princíp Huffmanovho kódovania

Pri Huffmanovom kódovaní sa dáta rozdeľujú do akéhosi binárneho stromu podľa toho koľkokrát sa jednotlivé symboly (zvukové frekvencie, farebné zložky) v daných dátach vyskytujú. Najfrekventovanejšie symboly v tomto súbore sa konvertujú do bitových reťazcov s najkratšou dĺžkou, symboly, ktoré sú menej zastúpené sa kódujú do reťazcov dlhších. Lepšie tento princíp pochopíme na príklade:
Povedzme, že obrázok sa skladá zo zložiek A, B, C, D a E, pričom najviac zastúpené sú zložky A a to 16-krát, zložka B sa nachádza v súbore 10×, C-8×, D-4× a E-7×. Jednotlivé časti si zoradíme podľa frekvencie výskytu pod seba (základňa stromu) a spojnicami spojíme spolu dve zložky s najnižším zastúpením v súbore, pričom hodnoty zložiek spočítame. Spojnice ohodnotíme hodnotami 0 a 1.
V konečnom dôsledku nám vznikne takýto strom:



Podľa toho ako sme jednotlivé zložky spájali vidíme binárne kódovanie jednotlivých znakov (postupnosť od konečného uzlu k počiatočnému uzlu) a zisťujeme, že platí to čo sme písali vyššie – zložky s najvyšším zastúpením majú najkratší bitový reťazec (všimnúť si môžeme taktiež ďalšiu zaujímavosť – žiaden kód nie je a ani nesmie byť predponou iného, takže pri dekódovaní súboru nedochádza k nepresnostiam):

A = 00, B = 10, C = 11, D = 011, E = 010

Takýmto spôsobom môžeme bez straty informácií zmenšiť súbory zhruba o 30-50 %.

Kompresia videa

Video má tu vlastnosť, že sa skladá z množstva za sebou idúcich snímkov (frameov, obrázkov), ktoré sú si viac či menej podobné Práve túto skutočnosť využíva MPEG kompresia, ktorá odstraňuje časovú

podobnosť. Pri MPEG kompresii sa nekódujú celé snímky, ale kóduje sa len zmena medzi jednotlivými obrazmi. Kodek rozdelí jednotlivé snímky videa na dva prípadne tri typy snímkov. Medzi kľúčové patria takzvané I-frames (*Intra coded image, I-VOP*), snímky, ktoré ku kompresii resp. dekompresii nepotrebujú žiaden iný snímok. Za I-frame nasleduje skupina niekoľkých snímkov, ktoré však už pre svoju kompresiu potrebujú jeden predchádzajúci obraz. Tieto snímky označujeme ako P-frames (predicted image, P-VOP). Kodeky založené na štandarde MPEG-4 ASP umožňujú používať aj tretí typ snímkov – B-frames (biderictionally interpolated image, B–VOP), ktoré sú odvodené ako od predchádzajúceho I alebo P snímku, tak i od nasledujúceho. Skupina snímkov medzi dvoma I-frames sa nazýva GOP (*Group of Pictures*) a nemala by obsahovať viac ako 12 snímkov - v prípade, že by GOP obsahovala viac ako 12 snímkov, nebolo by video kompatibilné s formátom DVD (PAL) a navyše by sa pri prehrávaní príliš dlho dekomprimovalo.

V súčasnej dobe sa používa niekoľko kompresných metód. Najstarší z nich (JPEG, H-261, MPEG-2) používajú skupiny pixelov (pokrývajúce určitú oblasť, napríklad 8 bitov x 8 bitov). Tieto metódy neanalyzujú aktuálnu štruktúru videa. Túto generáciu kompresorov je možné používať v prípade nižších rozlíšení, pokiaľ prenosová rýchlosť nie je menšia ako 64kbps.
Efektívnejšiu metódu kompresie ponúka druhá generácia algoritmov. Tieto algoritmy (napr. MPEG-4) vykonajú určitý druh analýzy obrazu pred uskutočnením odpovedajúcej kompresnej procedúry. Bolo zistené, že obraz je možné deliť na jednotlivé časti efektívnejším spôsobom. Obraz, ktorý sa skladá z pixelov s rovnakými vlastnosťami, je možné rozdeliť na špecifické skupiny. Tento pracovný postup výrazne zlepšuje proces kompresie. Aby bol dosiahnutý takýto analytický postup, bol použitý ľudsky chápajúci model (HVS - Human Vision System). Typické kódovacie zariadenie druhej generácie sa skladá z nasledujúcich modulov: členenie (segmentácia), odhad pohybu, kódovanie obrysov a textúr. Systém ponúka kompromis medzi rýchlosťou kompresie a kvalitou obrazu dosiahnutého po dekompresii.

Štandardy kompresie obrazu: H-261, H263

H-261 systém bol vytvorený pre ISDN siete so šírkou pásma 64 kbps. H-261 štandard umožňuje kompresnú rýchlosť od 100:1 do 2000:1, avšak čím vyšší kompresný pomer, tým nižšia kvalita dekomprimovaného obrazu. Kodek pracuje v dvoch režimoch: intra frame - INTRA a inter frame - INTER.
V priebehu INTRA kompresného režimu je snímok rozdelený do dvoch blokov o veľkosti 8x8 pixelov. Potom sú komprimované pomocou Discrete Cosines Transform (podobný ako JPEG).
V INTER režime sa ukladajú iba zmeny, ktoré nastali medzi nasledujúcimi snímkami. Musí však byť označené aspoň jeden krát, za 123 snímkov musí byť kompresia v INTRA režime uskutočnená.
Vývoj H.263 štandardu bol mienený pre zaistenie kompresie obrazového signálu pre siete s malou šírkou pásma, približne od 28.8 do 33.6 kbps.
H-263 je zmodernizovaná verzia H-261 štandardu. V porovnaní s pôvodnou verziou bola použitá half - pixelová kompenzácia pohybu miesto pixelovej kompenzácie pohybu. Bolo upustené od detekcie a prenosu korekcie chýb a boli integrované nové voliteľné režimy prevádzky.
Viac informácií nájdete na H-261, and H-263

MPEG-4 kompresia

MPEG-4 je najnovší štandard kompresie videa, ktorý sa rýchlo stáva populárnym, tiež pokiaľ je možný vo video streaming. V porovnaní s vyššie uvedenými štandardmi zapája efektívne metódy pre multimediálne údaje.
Nový prístup spočíva vo vzájomnom vzťahu údajových obsahov a úlohe ľudského vnímania, a vyňatí takzvaných audiovizuálnych objektov (AVO). Boli integrované riešenia umožňujúce rozšíriteľnosť zobrazenia údajov v mnohých rozmeroch, ktorými sú priestor, čas, kvalita a výpočtová komplexnosť.

Tento štandard bol nastavený pre mnoho typov telekomunikačných sietí a dokáže pracovať pri rýchlostiach od 64 do 1024 kbps.
Koncepcia MPEG-4 umožňuje prenos jednotlivého alebo niekoľkých audiovizuálnych AV objektov od vysielača k prijímaču, avšak skôr ako prenos začne, musí byť vykonaná počiatočná výmena informácií medzi kódovacím a dekódovacím zariadením. Týmto spôsobom sa priradí vhodný algoritmus a môžu byť stanovené nástroje nevyhnutné pre efektívne využitie spoja. Kódovacie zariadenie samozrejme chráni audiovizuálne objekty proti chybám.

Audio kompresia
Tok údajov z kompresie zvuku je podstatne menší ako u toku video údajov. Kompresia sa však normálne vykonáva za účelom minimalizácie celkovej šírky pásma.
Ľahkej redukcie šírky pásma spoja je možné dosiahnuť pomocou zníženia prenosovej rýchlosti (vzorkový kmitočet) a zníženie rozsahu amplitúdy.
V závislosti na type prenosu sa používa rôzne vzorkovanie:

Vzorkovací kmitočet	Aplikácia
8.00 kHz	Telefónia
16 kHz	Multimediálna komunikácia
22.05 kHz	Osobný počítač
32.00 kHz	Digitálne rádiové a televízne vysielanie
44.10 kHz	CD - Audio
48.00 kHz	DAT kazetové magnetofóny, HDTV
96.00 kHz	DVD

Typické vzorkové kmitočty

V telekomunikáciách je pre kódovanie hlasu používané 12 až 14-bitové kódovanie.
Okrem metód, ktoré spôsobujú určité zhoršenie kvality zvuku, sa používajú tiež kompresné metódy, ktoré prakticky neovplyvňujú pôvodnú podobu.
Príkladom takejto kompresie môže byť ADPCM *(Adaptive Differential Pulse Code Modulation)*, spočívajúcej v odhadnutí súčasného vzorku podľa predchádzajúceho vzorku, kvôli faktu, že u audio signálu je silná korelácia medzi nasledujúcimi vzorkami. ADPCM je bezstratová metóda.

Vlnná kompresia. Jedná sa o kompresiu založenú na použití vlnových algoritmov. Vlny sú matematickou funkciou berúcou hodnoty rôzne od nuly iba v určitých obmedzených intervaloch. Prostredníctvom odpovedajúcej matematickej konverzie môže byť týmto spôsobom uvedená každá funkcia.
Vstupný signál, ktorý chceme komprimovať, je filtrovaný od objektov, ľudským okom neviditeľných, potom je rozdelený na časti a každá z týchto častí je zastúpená odpovedajúcimi funkciami.
Táto metóda patrí k stratovým metódam, komprimované obrazy sú charakteristické rozmazanými okrajmi. Avšak v porovnaní s JPEG a MPEG metódami vlnná kompresia nespôsobuje skreslenie štruktúry objektu na obrazovke. Vlnné metódy sú stále vo vývoji a pravdepodobne sa stanú základom nových kompresných zariadení. Zaujímavý fakt je ten, že túto metódu použila FBI pre zavedenie systému pre rozpoznávanie odtlačkov prstov, nakoľko prešla testy s lepšími výsledkami ako JPEG.

Metóda vlnnej kompresie je dostupná, okrem MJPEG, aj pre užívateľov Hicap digitálneho nahrávacieho systému. Výrobca informuje, že pri použití MJPEG kompresie, 10GB voľného miesta na disku umožní zaznamenať 4.5 hodín živého materiálu, zatiaľ čo vlnná metóda zaberie čas 9 hodín. V každom prípade, pokiaľ potrebujeme lepšiu kvalitu, mali by sme použiť MJPEG kompresiu.

Význam kompresie

Kompresia (stlačenie, zhustenie) je snaha pomocou rôznych postupov zapísať dáta tak, aby zaberali menej miesta na záznamovom médiu. Zapísané dáta sa však musia dať previesť späť do pôvodného stavu alebo musia aspoň zachovať obsahový význam uložených dát (stratová kompresia). **Kompresia je teda iba iný spôsob uloženia údajov, tak aby zaberali menej miesta.**

Kompresia šetrí miesto. To v praxi znamená, že **na rovnaké záznamové médium sa nám zmestí viac údajov.** Napríklad ak pesničky skomprimujeme do formátu MP3 (MPEG 1 Layer 3), na CD sa nám zmestí až 12 CD pesničiek, ktoré nie sú komprimované. Rovnako, ak obrázky skomprimujeme do formátu JPEG, na CD sa nám zmestí oveľa viac obrázkov ako keby boli uložené vo formáte BMP.

Kompresia šetrí čas. Ak posielame komprimovaný súbor cez internet alebo pomalú sieť, **prenos bude rýchlejší,** ak údaje pred prenosom najprv skomprimujeme. Napríklad ak posielame pesničky vo formáte MP3, jej prenos bude 12 krát rýchlejší ako pri nekomprimovanej pesničke. Rovnako, ak umiestnime obrázok vo formáte JPEG na www stránku, tak jeho načítanie v prehliadači bude rýchlejšie ako keby sme ho tam umiestnili vo formáte BMP.

Okrem obrázkov a hudby môžeme komprimovať akékoľvek dáta. Na tento účel slúžia kompresné formáty ako napríklad **zip**, **gzip, tar** či **rar**. Kompresia sa však nedá použiť vždy. Niekedy údaje po komprimovaní zaberajú viac ako pôvodné dáta. O úspešnosti použitia kompresie hovorí **kompresný pomer.** Kompresný pomer je pomer skomprimovaných dát ku neskomprimovaným. **V prípade, že sa podarí súbor zmenšiť, je kompresný pomer vždy menší ako 1**. Ak by sme sa pokúšali napríklad komprimovať MP3 pomocou formátu zip, mohlo by sa stať, že kompresný pomer by bol väčší ako 1, teda výsledný ZIP súbor by bol väčší ako pôvodný súbor MP3. V takomto prípade kompresia nemá zmysel.

Princíp kompresie

Kompresia údajov je dvojakého typu:

- **Stratová** (lossy) - zachová sa iba významový obsah údajov
- **Nestratová** (lossless) - zachová sa pôvodný údaj bez akejkoľvek straty

Stratová kompresia je založená na odstránení nepodstatných údajov z dát (nepočuteľné zvuky, neviditeľné body, splývanie farieb...). **Údaje nie sú 100% rekonštruovateľné.**

Nestratová kompresia zmenší údaje tak, aby sa potom dali dať späť do pôvodného stavu. **Údaje sú 100% rekonštruovateľné.**

Pri kompresii sa využíva niekoľko typov algoritmov, najznámejšie sú:

- Huffmanov kód
- RLE
- LZW

Huffmanov kód je veľmi účinná kompresia používaná i formátmi JPEG a MPEG. Je založená na frekvenčnej analýze znakov (alebo skupín bitov). Princíp tohto kódovania je ten, že najčastejšie používané znaky sa kódujú menším počtom bitov ako normálne. Najskôr sa vytvorí tabuľka počtu výskytov všetkých znakov. Potom sú symboly usporiadané od najčastejšie použitých po najmenej často použité. Častejším symbolom sa potom priradí menší počet bitov ako menej častým.

Aritmetické kódovanie predstavuje ďalšiu efektívnu kompresnú metódu a je kandidátom na nahradenie Huffmanovho kódovania v rôznych aplikáciách, pretože dáva lepšie kompresné výsledky o 5 až 10%, aj keď za cenu náročných aritmetických operácií s veľkými reálnymi číslami. Je veľmi náročné na pamäť a výkon procesora. Aritmetické kódovanie nepracuje na princípe nahradzovania vstupného znaku špecifickým kódom. Namiesto toho, kódovaný vstupný tok znakov nahradí jedným reálnym číslom z intervalu $<0,1)$. Na začiatku uvažujeme celý tento interval. Ako sa správa predlžuje, spresňuje sa i výsledný interval a jeho horná a dolná hranica sa k sebe približujú. Čím je kódovaný znak pravdepodobnejší, tím sa interval zúži menej a k zápisu dlhšieho intervalu stačí menej bitov.

Spôsob aritmetického kódovania si ukážeme na príklade správy "baca".

Znak	Pravdepodobnosť
a	0.5
b	0.25
c	0.25

Pravdepodobnosti výskytu znakov sú nám známe a potrebujeme prideliť každému znaku určitý interval z rozsahu $<0,1)$ na základe pravdepodobnosti jeho výskytu. Pritom nieje jedno, ktorému intervalu bude pridelený daný znak. Naše 3 znaky budú zaradené do jednotlivých intervalov nasledovne.

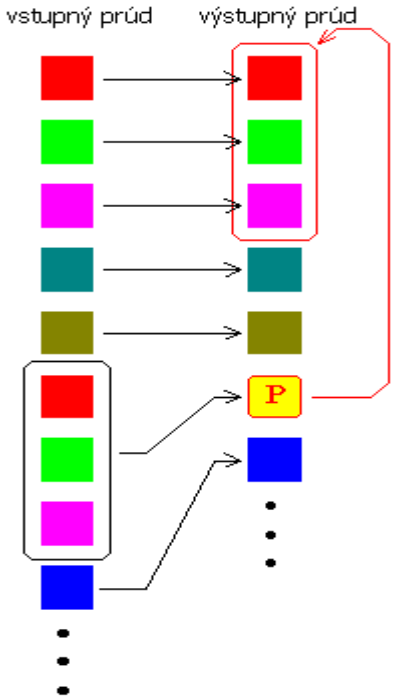
Znak	Pravdepodobnosť	Interval
a	0.5	0.00 - 0.50
b	0.25	0.50 - 0.75
c	0.25	0.75 - 1.00

Nakoľko, každý interval je vľavo uzavretý a vpravo otvorený tak napr. c bude v intervale 0.75 - 0.9999... . Dôležitý význam pri aritmetickom kódovaní má prvý kódovaný znak. Keď kódujeme správu "baca" , prvý znak je "b". Začiatočný znak nám určuje, že správa bude zakódovaná ako číslo, ktoré sa nachádza v intervale odpovedajúceho prvému znaku. V našom prípade môžeme očakávať číslo v rozsahu 0.5 - 0.75. Takto sme zakódovali prvý znak . Rozsah sa nám zmenšil na 0.5 - 0.75. Ďalší znak je "a" a prináleží mu interval 0.0 - 0.5. V rozsahu 0.5 - 0.75 bude časť 0% - 50% (interval pre "a" je 0.0 - 0.5) z tohto rozsahu reprezentovať znak "a". Tím sa nám rozsah zase zmenšil na rozmer 0.5 - 0.625. Analogickým spôsobom pokračujeme v delení intervalu ďalej.

Slovníkové metódy

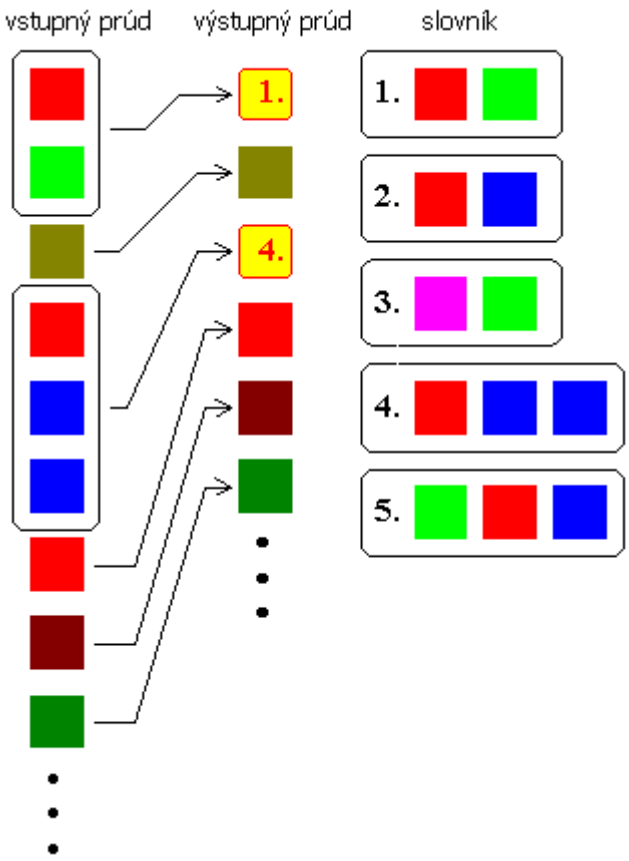
Tieto kompresné metódy využívajú toho, že v komprimovaných dátach sa nachádzajú opakujúce sa sekvencie dát. Všetky slovníkové metódy môžeme rozdeliť do dvoch základných skupím.

Metódy z prvej skupiny sa pokúšajú nájsť, či práve komprimovaná sekvencia znakov sa už skôr nevyskytovala vo vstupnom prúde dát a namiesto jej opakovania, zapíšeme na výstup odkaz na jej predchádzajúci výskyt v prúde dát. Princíp je zobrazený na obrázku.



Všetky algoritmy z tejto skupiny sú založené na algoritme publikovanom v roku 1977 pánmi Abrahamom Lempelom a Jakobom Zivom s názvom LZ77. Ďalšie modifikácie tohto algoritmu nesú názvy LZSS, LZB, LZH,...

Druhá skupina algoritmov si vytvára slovník reťazcov, ktoré sa vyskytli vo vstupnom prúde dát. Pokiaľ kompresor narazí na reťazec, ktorý už bol umiestnený v slovníku (resp. vyskytoval sa v predchádzajúcom prúde dát), zapíše na výstup kód, ktorý reprezentuje reťazec uložený v slovníku. Najlepšie princíp tejto metódy ilustruje obrázok.



Metódy z tejto skupiny majú základy v algoritme, ktorý v roku 1978 publikovali Lempel a Ziv a je označovaný ako LZ78. V roku 1984 modifikoval metódu LZ78 pán Terry Welch a nesie označenie LZW.

Prúdové kódovanie (RLE - Run Length Encoding)

Metóda prúdového kódovania (RLE) je komprimačnou metódou, ktorá fyzicky zredukuje akýkoľvek typ sekvencie opakujúcich sa znakov, hneď ako ich počet dosiahne predpísaný počet výskytov. Táto metóda je nestratová, symetrická a obyčajne sa vyznačuje veľkou rýchlosťou kódovania za cenu pomerne nízkeho kompresného pomeru. Kompresný pomer však záleží na spracovávaných dátach a môže byť v niektorých prípadoch naopak veľmi vysoký.Kompresia RLE je použitá napr. v grafických formátoch PCX, PSD (Adobe Photoshop), BMP (Windows bitmap), TGA (Targa) v protokoloch IBM 3780 BISYNC, MNP triedy 5 a iných.
Kompresoru je potrebné zadefinovať špeciálny znak indikujúci komprimáciu. Najvhodnejšie je za indikátor kompresie zvoliť znak, ktorý sa vo vstupnom prúde dát nevyskytuje. Použitie metódy prúdovej komprimácie vedie k tomu, že postupnosť opakujúcich sa znakov bude zakódovaná do nasledovného tvaru:

I_k - indikátor kompresie **X** - ľubovoľný znak **P_o** - počet opakujúcich sa znakov **X**

Pretože pre zakódovanie sekvencie je potrebné použiť tri znaky, je zrejmé, že metóda bude efektívna iba pri komprimácii štyroch a viac opakujúcich sa znakov.

Príklad:

```
Pôvodné dáta      : Ann$b4155555555PklAAAAAA
Komprimované dáta: Ann$b41Ik58PklIkA6

55555555  <-zakódujem->  Ik 5 8
AAAAAA    <-zakódujem->  Ik A 6
                        [Ik X Po]
```

Run-Lenght Encoding (RLE) je kompresia vhodná na použitie v reálnom čase. To znamená, že vytváranie výsledných komprimovaných dát je také rýchle, že to používateľ systému ani nepocíti. Táto kompresia bola v minulosti použitá i na ukladanie obrázkov nakreslených skicárom do formátu PCX. Kompresia funguje na binárnej úrovni - pracuje rovno s bitmi. Princíp kompresie je ten, že sa striedavo zapisuje počet jednotiek a počet núl, ktoré sa zapíšu v dvojkovej sústave.

Príklad:

```
 vstupný reťazec: 00011000000111110000110001111 (28 bitov)
       počty núl a jednotiek: 3 2 6  5 3 2 3 4
       výstupný reťazec  011 010 110 101 011 010 011 100 (24 Bitov)
ušetrili sme 4 bity
```

LZW kompresia je skratkou podľa jej autorov (Abraham **L**empel, Jacop **Z**iv, Terry **W**elch). Táto kompresia sa používa napríklad vo formáte GIF. Kompresia je založená na vytváranie dynamického slovníka "slov" (skôr ide o ľubovoľné spojenie znakov), ktorým sa priraďujú čísla podľa ich prvého výskytu. Na začiatku sú v slovníku iba slová s dĺžkou jedna. Potom pribúdajú slová s dĺžkou dva a potom dlhšie a dlhšie. To znamená, že zo začiatku sú nahrádzané iba malé postupnosti, ale neskôr sa jedným číslom môže nahradiť i dlhé slovo.

Príklady popisov programov na kompresiu dát/súborov

7-Zip je voľne distribuovateľný komprimačný program s vlastným formátom 7z, ktorý je vo väčšine prípadoch účinnejší ako formát RAR či ACE, ZIP. Má vysoký kompresný pomer, schopnosť použiť rôzne metódy kompresie (LZMA, PPMD, BCJ, BCJ2,BZip2, Deflare), zaheslovanie/šifrovanie/konverzia, samorozbaľovacie archívy (EXE). Hlavný formát 7z je o 30-70% lepší ako formát ZIP. Do formátu ZIP a GZIP komprimuje 7-Zip o 2-10% lepšie ako PKZip či WinZip.
Podpora formátov:
Kompresia/dekompresia: 7z, ZIP, GZIP, BZIP2, TAR
Dekompresia: RAR, CAB, ARJ, LZH, CHM, Z, CPIO, RPM a DEB, ISO, SWM, TBZ2, WIM. Obsahuje Slovenčinu/Češtinu.

Obľúbený komprimačný nástroj, pričom samotný RAR súbor má vysoký komprimačný pomer v porovnaní s inými komprimačnými formátmi (ZIP, ARJ a LZH). Do 1 archívu typu

RAR je možno archivovať 9 miliónov TB dát. Poradí si tiež aj s ZIP, CAB, ARJ, LZH, ACE, UUE, TAR a GZ súbormi.

Šifrovanie

Šifrovanie je kryptografické pretváranie údajov, ktorého výsledkom je šifrovaný text. Takýto text sa tretím osobám javí ako náhodný reťazec znakov, z ktorého nemožno vyčítať užitočnú informáciu.

Medzi najznámejšie kryptografické algoritmy patria: DES, RSA, MD5, SHA, IDEA. Korene kódovania informácií siahajú do hlbokej minulosti ľudských dejín. Až rozvoj matematiky priniesol ovocie v podobe vzniku pomerne silných kryprografických algoritmov. Najpoužívanejšími metódami šifrovania sú:

- symetrické šifrovanie,
- asymetrické šifrovanie,
- hashovacie funkcie.

Treba ešte pripomenúť, že dôvodom šifrovania nemusí byť iba snaha utajiť informáciu. Stačí si spomenúť Morseovú abecedu. Namiesto písmen používa bodky a čiarky (krátke a dlhé signály).
Cieľom Morseovej abecedy nie je správu utajiť, ale iba efektívne sprostredkovať.

Čo je vlastne šifrovanie?
Hneď na úvod by som chcel upozorniť na časté miešanie pojmov kódovanie a šifrovanie. Nie, nie je to to isté, hoci majú veľa spoločného. **Kódovanie** má za cieľ umožniť ukladanie informácií (zvyčajne ich číselnou reprezentáciou) či uľahčiť ich prenos, **šifrovanie** sa snaží o „znečitateľnenie“ dokumentu pre toho, komu nie je určený – ide teda o utajenie obsahu. Kým pri dekódovaní zakódovanej správy nám stačí poznať (spravidla verejne známy) postup, na dešifrovanie zašifrovanej správy musíme poznať i akési heslo či kľúč.

Symetrické šifrovanie
Symetrické šifrovanie je v kryptografii klasika a práve to, čo si pod pojmom šifrovanie predstaví väčšina ľudí. Čo je preň charakteristické? Používa sa **jediný kľúč**, ktorým možno dokument zašifrovať i dešifrovať. Niečo podobné ako trezor – každý, kto má kľúč, môže si ho otvoriť a prezrieť dokumenty uložené v ňom, ako aj vkladať ďalšie dokumenty. Ak sa niekomu kľúč dostane do rúk, mohol by si ho „skopírovať“ a získať tak prístup kedykoľvek. Možno ste niektoré jednoduché šifrovanie už kedysi používali pri posielaní správičiek spolužiakom na nudných hodinách. Uveďme si pár príkladov.
Veľmi jednoduchá šifra na detské hry by mohla vyzerať napríklad takto – každé písmenko „zväčšíme“ o 2, teda namiesto A napíšeme C, namiesto B bude D atď. Text AHOJ by po zašifrovaní bol CJQL. Táto šifra sa nazýva **posuvná šifra** (shift cipher). Ak zašifrovanú správu pošleme príjemcovi, tomu stačí vedieť, aký postup sme použili, a poznať správny kľúč. Čo je v tomto prípade kľúčom? Je to číslo 2, o ktoré treba zase „zmenšiť“ každé písmenko zašifrovanej správy.
A čo ak sa zašifrovaná správa dostane do rúk nepovolanej osobe, ktorá nepozná kľúč? Ak bude mať záujem dozvedieť sa obsah správy, asi mu v tom nezabránime, je veľmi ľahké túto jednoduchú šifru nalomiť i použitím toho najnevýhodnejšieho útoku, tzv. útoku hrubou silou

(brute-force attack), ktorý vlastne predstavuje skúšanie všetkých možností. No koľko je tých všetkých možností? Ak budeme predpokladať iba písmenká bez diakritiky, máme toľko možností, koľko je písmen v abecede (26), vlastne o jednu menej, pretože „zvýšenie o 0“ nemá význam.

Poznámka: Abecedu pokladáme za cyklickú, teda po písmene Z nasleduje opäť A – napríklad „zvýšenie o 25“ je vlastne to isté, ako „zníženie o 1“, pretože z A sa stane Z, z B bude A, z C zase B atď.

Skúsme uvedenú šifru zlepšiť, a to tak, že nebudeme písmenká konštantne „zvyšovať“ (vlastne posúvať abecedu), ale ich úplne rozhádžeme. Napríklad namiesto A bude R (A -> R), B -> D, C -> K, D -> Z a pod. Takáto šifra sa nazýva **substitučná šifra** (substitution cipher) a spomenutá posuvná šifra, ktorou sme sa zaoberali, je len jej podmnožinou. Čo je vlastne kľúčom pri substitučnej šifre? Už to nie je jediné číslo, ale celé usporiadanie, teda tzv. permutácia – ide o usporiadanú množinu 26 písmen, ktorá by sa v našom príklade začínala [R, D, K, Z...]. Matematicky zdatnejším čitateľom je zrejmé, že existuje 26! rôznych kľúčov, čo je asi 4 × 1026. Je to veľmi veľa, a preto sa táto šifra na prvý pohľad zdá pomerne silná. Lenže pozor, na šifru vôbec netreba útočiť hrubou silou! Predpokladajme, že poznáme prvé slovo pôvodného textu, napr. AHOJ. Potom už poznáme zámenu 4 písmen. Ak ich dosadíme v celom texte na príslušné miesta, môžeme začať postupne hádať – doplňať písmenká tam, kde sa akosi „hodia“. Najhoršie je teda začať, ale i s tým sa dá vyrovnať.
Pre každý jazyk možno vytvoriť určitú tabuľku pravdepodobnosti výskytu jednotlivých písmen v bežnom texte, napr. v anglickom jazyku je najčastejším písmenkom E (p = 0,127 s veľkým náskokom), takže ak máme pred sebou zašifrovaný anglický text, zistíme si najčastejšie sa vyskytujúce písmeno v zašifrovanom texte a môžeme ďalej predpokladať, že vyjadruje písmeno E. Podobne sa dajú podľa výskytu usporiadať i dvojice a trojice písmen (v angličtine dvojice TH, HE, IN..., trojice THE, ING, AND...). Po nacvičení postupu dokážeme text dešifrovať za menej ako hodinu. Čím je dlhší, tým lepšie pre útočníka. A ak využijeme pomoc počítača, ktorý nám veľmi rýchlo vie vykonať štatistickú analýzu a „dosádzať“ písmenká, šifru nalomíme prakticky hneď.
Na ukážku (a získanie predstavy) uvedené príklady stačia. Keby sme spojili viacero takýchto jednoduchých šifier (text zašifrujeme najskôr jednou, potom tento zašifrovaný text opäť zašifrujeme, ale už inou šifrou – kľúčom by bola postupnosť všetkých použitých čiastkových kľúčov), získame už pomerne silnú šifru, ktorú je veľmi obťažné nalomiť. Takto fungujú i známe symetrické šifry **DES** (Data Encryption Standard) a **AES** (Advanced Encryption Standard).
Teraz, keď už vieme, o čo ide pri symetrickom šifrovaní, zamyslime sa nad jeho používaním. Predpokladajme, že máme k dispozícii veľmi silnú šifru, ktorú prakticky nemožno nalomiť bez znalosti kľúča, a predstavme si situáciu, keď chce Alica poslať Bobovi tajnú správu. Ak **obaja poznajú kľúč**, nie je nijaký problém – Alica ním správu zašifruje, pošle ju Bobovi, Bob si ju týmto kľúčom zase dešifruje. Pridajme ešte ďalšiu osobu – Oskara, ktorý sa chce dozvedieť obsah prenášanej správy. Ako správny záškodník, Oskar má prístup ku komunikačnému kanálu, ktorým Alica komunikuje s Bobom (týmto kanálom môže byť klasická pošta, telefónna linka, kábel počítačovej siete a podobne) – teda dokáže všetko „odpočúvať“. Čo teda zachytí Oskar? Iba bezcennú podobu správy – prenáša sa predsa zašifrovaná. Ak majú Alica i Bob bezpečne uschovaný kľúč, nemusia sa báť, že by si správy čítal i niekto iný.

Čo sa však stane, ak chce Alica poslať správu Bobovi, no ten nemá Alicin kľúč? Práve tu je slabina symetrického šifrovania. Alica **nemôže kľúč poslať** verejným (odpočúvaným)

kanálom, ktorý sleduje Oskar. Lenže každý komunikačný kanál je potenciálne odpočúvaný – nemôžu vylúčiť, že Oskar odpočúva, a teda to musia predpokladať. Ak nemajú nijaký „zaručene bezpečný“ komunikačný kanál, nezostáva im nič iné, iba sa stretnúť a osobne si odovzdať kľúč – potom už budú môcť bezpečne komunikovať i v budúcnosti. Osobné stretnutie a odovzdanie kľúča sa nám síce môže javiť ako celkom dobrý nápad, ale čo ak sa nachádzajú na druhom konci sveta?

Asymetrické šifrovanie

Veľmi významným krokom v kryptografii bolo vytvorenie postupov asymetrického šifrovania. Čím sa líši od symetrického? Namiesto jediného kľúča, ktorý slúži zároveň na šifrovanie i dešifrovanie, sa tu používa **dvojica kľúčov** (key pair). Ak jedným z nich správu zašifrujeme, možno ju dešifrovať iba tým druhým. Pochopiteľne, tieto dva kľúče sú akosi matematicky zviazané, teda patria k sebe a boli spolu vygenerované. No platí, že znalosť jedného z dvojice kľúčov nepostačuje na získanie toho druhého.

Nazvime zatiaľ jeden kľúč „šifrovací“, ten druhý „dešifrovací“. Keďže zašifrovanú správu by mal vedieť dešifrovať dešifrovacím kľúčom iba ten, pre koho je určená, a nikto iný, potom by sme mohli dešifrovací kľúč označiť ako **súkromný**, a keďže zašifrovať správu pre nejakého príjemcu šifrovacím kľúčom by mal mať možnosť každý, šifrovací kľúč označíme ako **verejný**. Ďalej už budeme používať iba tieto dva pojmy, teda verejný kľúč slúži na šifrovanie (je voľne k dispozícii pre verejnosť), súkromný kľúč sa použije na dešifrovanie (jeho vlastník si ho drží v bezpečí).

Vráťme sa k Alici a Bobovi. Alica chce poslať Bobovi tajnú správu. Prečítať ju môže iba Bob, teda má byť dešifrovateľná iba jeho súkromným kľúčom, z čoho ďalej vyplýva, že ju Alica musí zašifrovať jeho verejným kľúčom. Aký bude teda postup?

1. Bob si vygeneruje svoju dvojicu kľúčov. Jednu jeho časť zverejní (verejný kľúč), druhú si ponechá (súkromný kľúč).
2. Alica si skopíruje Bobov verejný kľúč.
3. Alica zašifruje správu Bobovým verejným kľúčom.
4. Alica pošle zašifrovanú správu Bobovi.
5. Bob prijatú správu dešifruje svojím súkromným kľúčom.

Treba si uvedomiť jednu zaujímavosť – keďže je správa zašifrovaná Bobovým verejným kľúčom, dokáže ju dešifrovať jedine on, teda po zašifrovaní ju už nedokáže dešifrovať ani Alica, ktorá ju zašifrovala! (Dá sa však očakávať, že to nebude potrebovať.)

A čo na to zvedavý Oskar? Komunikačným kanálom najskôr putuje verejný Bobov kľúč (ktorý je však normálne k dispozícii pre všetkých, teda i pre Oskara), potom už len zašifrovaná správa. Čo sme teda získali asymetrickým šifrovaním? Odstránili sme závažný nedostatok symetrického šifrovania – nie je potrebné, aby sa Alica a Bob osobne stretli a odovzdali si kľúč.

Pre úplnosť dodajme, že ak chce Bob odpísať Alici, musí si nahrať jej verejný kľúč a správu zašifrovať ním.

ko funguje šifrovanie na internete V prvej časti sme si vysvetlili základné princípy symetrického a asymetrického šifrovania a teraz sa zamyslime nad tým, ako prebieha prenos informácií napríklad pri navštívení internetovej banky.

Hoci by to malo byť každému zrejmé, radšej pripomeniem, že všetky údaje, ktoré sa posielajú po internete prostredníctvom klasického HTTP protokolu (čiže internetové stránky, obsah formulárov na nich, obrázky), či ostatných protokolov, napr. POP3 (na prevzatie e-mailových správ zo servera), **sú prenášané v čitateľnej podobe** (nešifrovane), a teda ich môže získať prakticky ktokoľvek. Vzhľadom na to, že programové nástroje slúžiace na zachytávanie

citlivých informácii sú ľahko dostupné, nemusí ísť ani o nijakého počítačového piráta (atraktívnejšie znie hackera) - dokáže to, žiaľ, hocikto.

Vaše heslá, prístupové kódy, ako aj samotné texty správ (je jedno, či e-mailov, SMS-iek, diskusných príspevkov...) už možno dávno skončili v rukách niekoho ďalšieho a vy o tom ani neviete... Ak pristupujete na internet kdesi v škole, či na internáte, môžete si tým byť takmer istí. Ale nebuďme pesimisti, pozrime sa radšej, ako tomu predísť. No predsa šifrujme! Nápad to nie je zlý, ale ako na to? Našťastie je to jednoduchšie, ako sa na prvý pohľad možno zdá, veľa námahy nás to stáť nebude - no je na to potrebná podpora zo strany cieľového servera.

Okrem klasického HTTP protokolu existuje aj jeho šifrovaný variant - **HTTPS**, môžete sa v tejto súvislosti stretnúť aj so skratkou **SSL (Secure Socket Layer)**, čo je vlastne prídavná šifrovacia vrstva, ktorá pod obyčajným HTTP protokolom vytvorí zabezpečený kanál. Ak ho podporuje váš obľúbený server, na ktorý musíte posielať nejaké neverejné údaje (zvyčajne heslo), stačí vám do internetového prehliadača namiesto zvyčajného *http://www.nieco.sk* napísať ***https***://www.nieco.sk - teda pridať písmenko S k názvu protokolu. Šifrované spojenie ponúka každý lepší freemail, či portál, ktorý to myslí s bezpečnosťou svojich používateľov vážne - príkladom buď POBOX, Orangeportal, E-zones. Každý internetový prehliadač vám nejakým spôsobom dá najavo, že sa nachádzate na zabezpečenej stránke - napríklad Internet Explorer túto skutočnosť vyjadruje žltým zámkom v pravej časti stavového riadku. Ak naň kliknete, zobrazia sa vám detailné informácie o zabezpečení, okrem iného aj „sila“ šifrovania – 40-bitové je veľmi slabé, je možné ho v relatívne krátkom čase prelomiť, no 128-bitové šifrovanie je v súčasnosti úplne bezpečné. Niektoré servery nemajú šifrované všetky prenosy, ale iba proces prihlasovania sa (teda login a heslo sa prenesú zašifrovane, ale ostatné texty v čitateľnom tvare).

SSL dokáže kryť nielen protokol HTTP, ale i poštové protokoly POP3, SMTP a iné. Dôležité je šifrovať všetko, kde sa prenášajú heslá (napr. vždy pri sťahovaní pošty z poštového programu cez POP3 či IMAP4).

Skúsme sa teraz zamyslieť, aký druh šifrovania by bolo možné použiť. Symetrické či asymetrické?

Na **symetrické šifrovanie** je potrebné mať spoločný kľúč. Ak ideme komunikovať s internetovou bankou, musí mať rovnaký kľúč, ako máme my, no nesmie ho mať nikto iný. Samozrejme, že po internete ho prenášať nemôžeme. Povedzme, že by sme si do banky osobne zašli zobrať nejaký ten „kľúč“. Ale pôjdeme ho brať i ku každému prevádzkovateľovi freemailu, či dokonca ku každému, kto prevádzkuje akékoľvek stránky s možnosťou šifrovaného prístupu? Zdá sa, že tadiaľto cesta nevedie.

A čo **šifrovanie asymetrické**? Banka vystaví svoj verejný kľúč, my si ho nahráme (resp. nahrá si ho náš internetový prehliadač) a zašifrujeme ním odosielané prihlasovacie údaje (opäť to za nás vykoná prehliadač). Zatiaľ je to všetko fajn. Lenže ako bude banka posielať odpovede nám? Asi by nám vadilo, keby si informácie o stave nášho účtu, či o transakciách mohol prezerať i niekto iný - takto si bezpečnú transakciu nepredstavujeme. Aby nám banka mohla odpovedať cez šifrované spojenie, museli by sme aj my vystaviť svoj verejný kľúč. To sa už zdá byť reálnejšie, no netreba zabúdať na fakt, že nie každý aktívne používa šifrovanie, teda nie každý má svoj vlastný kľúč. Prehliadač si však môže nejaký ten dočasný kľúč vygenerovať, slúžiť bude len na jedno použitie. A pritom by nemusel byť ani asymetrický (matematicky náročnejší, a teda pomalší), veď aj obyčajný symetrický kľúč sa môže banke

poslať zašifrovane...

Dospeli sme teda ku **kombinácii asymetrického a symetrického šifrovania**, zjednodušený postup bezpečnej transakcie by mohol byť takýto:

1. prehliadač si náhodne vygeneruje dočasný symetrický kľúč;
2. prehliadač si nahrá verejný asymetrický kľúč servera (napr. banky);
3. prehliadač zašifruje dočasný kľúč verejným kľúčom servera a pošle ho serveru;
4. ďalšia komunikácia už môže byť symetricky šifrovaná dočasným kľúčom.

ifrovanie e-mailových správ v programe Microsoft Office Outlook 2007 umožňuje chrániť osobné údaje správy konverziou obyčajného a čitateľného textu na zašifrovaný text. Správu môže dešifrovať len príjemca s osobným kľúčom (osobný kľúč: utajený kľúč uložený v počítači odosielateľa, ktorý odosielateľ používa na digitálne podpisovanie správ pre príjemcov a na odšifrovanie (odomknutie) správ od príjemcov. Osobné kľúče by mali byť chránené heslom.), ktorý zodpovedá verejnému kľúču (verejný kľúč: kľúč, ktorý odosielateľ poskytne príjemcovi, aby príjemca mohol overiť podpis odosielateľa a potvrdiť, že správa nebola zmenená. Príjemcovia používajú verejný kľúč tiež na zašifrovanie (uzamknutie) e-mailových správ odosielateľovi.) použitému na zašifrovanie správy.

Poznámka. Šifrovanie správy je iný proces ako digitálne podpisovanie správy.

ŠIFROVANIE
-časté miešanie pojmov kódovanie a šifrovanie.
-kódovanie má za cieľ umožniť ukladanie informácií (zvyčajne ich číselnou reprezentáciou) či uľahčiť ich prenos, šifrovanie sa snaží o „znečitateľnenie“ dokumentu pre toho, komu nie je určený – ide teda o utajenie obsahu.
-na dešifrovanie zašifrovanej správy musíme poznať i akési heslo či kľúč.

SYMETRICKÉ
-to, čo si pod pojmom šifrovanie predstaví väčšina ľudí.
-používa sa jediný kľúč, ktorým možno dokument zašifrovať i dešifrovať.
-niečo podobné ako trezor – každý, kto má kľúč, môže si ho otvoriť a prezrieť dokumenty uložené v ňom, ako aj vkladať ďalšie dokumenty.
-ak sa niekomu kľúč dostane do rúk, mohol by si ho „skopírovať“ a získať tak prístup kedykoľvek.
-Napr.: Veľmi jednoduchá šifra na detské hry by mohla vyzerať napríklad takto – každé písmenko „zväčšíme“ o 2, teda namiesto A napíšeme C, namiesto B bude D atď. Text AHOJ by po zašifrovaní bol CJQL.

Táto šifra sa nazýva posuvná šifra (shift cipher). Ak zašifrovanú správu pošleme príjemcovi, tomu stačí vedieť, aký postup sme použili, a poznať správny kľúč. Čo je v tomto prípade kľúčom? Je to číslo 2, o ktoré treba zase „zmenšiť“ každé písmenko zašifrovanej správy.

-čo ak sa zašifrovaná správa dostane do rúk nepovolanej osobe, ktorá nepozná kľúč? Ak bude mať záujem dozvedieť sa obsah správy, asi mu v tom nezabránime, je veľmi ľahké túto jednoduchú šifru nalomiť i použitím toho najnevýhodnejšieho útoku, tzv. útoku hrubou silou (brute-force attack), ktorý vlastne predstavuje skúšanie všetkých možností.
-koľko je tých všetkých možností? Ak budeme predpokladať iba písmenká bez diakritiky, máme toľko možností, koľko je písmen v abecede (26), vlastne o jednu menej, pretože „zvýšenie o 0“ nemá význam.

-môžme uvedenú šifru zlepšiť, a to tak, že nebudeme písmenká konštantne „zvyšovať“ (vlastne posúvať abecedu), ale ich úplne rozhádžeme. Napríklad namiesto A bude R (A -> R), B -> D, C -> K, D -> Z a pod. Takáto šifra sa nazýva substitučná šifra (substitution cipher) a spomenutá posuvná šifra, ktorou sme sa zaoberali, je len jej podmnožinou. Čo je vlastne kľúčom pri substitučnej šifre? Už to nie je jediné číslo, ale celé usporiadanie, teda tzv. permutácia – ide o usporiadanú množinu 26 písmen, ktorá by sa v našom príklade začínala [R, D, K, Z...]. Existuje 26! rôznych kľúčov, čo je asi 4 × 1026.

Je to veľmi veľa, a preto sa táto šifra na prvý pohľad zdá pomerne silná. Lenže pozor, na šifru vôbec netreba útočiť hrubou silou! Predpokladajme, že poznáme prvé slovo pôvodného textu, napr. AHOJ. Potom už poznáme zámenu 4 písmen. Ak ich dosadíme v celom texte na príslušné miesta, môžeme začať postupne hádať – dopĺňať písmenká tam, kde sa akosi „hodia“.

-pre každý jazyk možno vytvoriť určitú tabuľku pravdepodobnosti výskytu jednotlivých písmen v bežnom texte, napr. v anglickom jazyku je najčastejším písmenkom E (p = 0,127 s veľkým náskokom), takže ak máme pred sebou zašifrovaný anglický text, zistíme si najčastejšie sa vyskytujúce písmeno v zašifrovanom texte a môžeme ďalej predpokladať, že vyjadruje písmeno E. Podobne sa dajú podľa výskytu usporiadať i dvojice a trojice písmen (v angličtine dvojice TH, HE, IN..., trojice THE, ING, AND...).

Po nacvičení postupu dokážeme text dešifrovať za menej ako hodinu. Čím je dlhší, tým lepšie pre útočníka. A ak využijeme pomoc počítača, ktorý nám veľmi rýchlo vie vykonať štatistickú analýzu a „dosádzať“ písmenká, šifru nalomíme prakticky hneď.

-keby sme spojili viacero takýchto jednoduchých šifier (text zašifrujeme najskôr jednou, potom tento zašifrovaný text opäť zašifrujeme, ale už inou šifrou – kľúčom by bola postupnosť všetkých použitých čiastkových kľúčov), získame už pomerne silnú šifru, ktorú je veľmi obťažné nalomiť. Takto fungujú i známe symetrické šifry DES (Data Encryption Standard) a AES (Advanced Encryption Standard).

-špičkovým algoritmom je v súčasnosti algoritmus IDEA (International Data Encryption Algorithm) publikovaný v roku 1990. IDEA s 128-bitovým kľúčom je dnes považovaný za bezpečný.

ASYMETRICKÉ
-veľmi významným krokom v kryptografii bolo vytvorenie postupov asymetrického šifrovania.
-namiesto jediného kľúča, ktorý slúži zároveň na šifrovanie i dešifrovanie, sa tu používa dvojica kľúčov (key pair). Ak jedným z nich správu zašifrujeme, možno ju dešifrovať iba tým druhým. Pochopiteľne, tieto dva kľúče sú akosi matematicky zviazané, teda patria k sebe a boli spolu vygenerované. No platí, že znalosť jedného z dvojice kľúčov nepostačuje na získanie toho druhého.
-najpopulárnejším asymetrickým šifrovacím algoritmom je algoritmus RSA (River, Shamir, Adelman), ktorý sa používa s rôznou dĺžkou kľúča, napr. 1024 bitov. Tento algoritmus sa nepodarilo ešte nikomu zlomiť a je považovaný za dostatočne bezpečný. RSA algoritmus sa môže použiť napríklad na prenos tajného DES kľúča druhej strane (tento DES kľúč sa vygeneruje na počítači odosielateľa náhodným spôsobom) a v ďalšom sa potom už pre vzájomnú komunikáciu používa algoritmus DES, pretože ten, ako už bolo spomenuté, je rýchlejší (t.j. proces šifrovania a dešifrovania trvá omnoho kratší čas ako pri asymetrickom

šifrovaní). Toto však platí iba pre jedno spojenie (communication session). Pri nadviazaní ďalšieho spojenia sa vygeneruje nový DES kľúč, pošle sa druhej strane zašifrovaný algoritmom RSA, atď.

Šifra

Šifrovanie používame na ochranu osobných údajov, čiže na zamedzenie prístupu k informácii osobe nato nepovolanej.

Šifrovanie delíme na symetrické a nesymetrické.

-**Symetrické šifrovanie** zakóduje informáciu I podľa daného kľúča K a dostaneme zakódovanú správu S. Pokiaľ chceme správu S spätne dekódovať musíme poznať kľúč K použitý pri kódovaní správy. Tu nastáva problém s prenosom kľúča. Kľúč sa totiž musí preniesť na nejakom médiu medzi odosielateľom a príjemcom správy. Toto môže byť nebezpečné z dôvodu odpočúvania pri prenose cez elektronický kanál, fyzický prenos je zase zdĺhavý. Tieto problémy rieši nesymetrické šifrovanie.

-**Nesymetrické šifrovanie** zakóduje informáciu I podľa kľúča K1(často nazývaný verejný kľúč- public key) ktorý odosielateľ získa od príjemcu správy. U príjemcu je správa spätne dekódovaná pomocou kľúča K2(súkromný kľúč- private key) ku ktorému má prístup iba príjemca správy. To znamená, že zakódovanú správu odosielateľ po zakódovaní nedokáže spätne dekódovať, pretože nevlastní kľúč K2, ktorý je potrebný na dekódovanie. Verejný a súkromný kľúč sú jedinečné /k jednému verejnému existuje iba jeden súkromný/ a jeden z druhého sa nedajú nijakým spôsobom určiť.

Dnes je **šifrovanie** zabezpečené **na nesymetrickej úrovni už samotnými poštovými klientmi** ako sú napríklad Mozilla Thunderbird, či Outlook (pozn. správa je šifrovaná symetricky, nesymetricky je šifrovaný iba šifrovací kľúč správy – dôvodom je časová náročnosť asymetrického šifrovania) . **Šifrovanie dát pri komunikácii na webe** zabezpečuje napríklad protokol **HTTPS** (čo je šifrovaný HTTP) alebo protokol **SSH**. Takéto šifrovanie sa používa najmä pri poskytovaní osobných údajov (kódy PIN, čísla účtov...) vzdialeným serverom.

Šifrovaním ale nerozumieme iba problematiku utajovania obsahu prenesených dát, **ale aj integritu dát**. To znamená, že pojem šifrovanie neobsahuje len ochranu údajov pred nepovolanými osobami, ale aj ich ochranu pred poškodením alebo úmyselným pozmenením dát. Toto zisťovanie chybovosti nazývame jednosmernou alebo **hašovacou funkciou**. Tieto funkcie sú neodlúčiteľnou časťou ochrany osobných údajov ako napríklad hesiel. Princíp je jednoduchý. Používateľ zadá svoje heslo pomocou hašovacej funkcie sa vytvorí jeho message digest (výstup z hašovacej funkcie) to s týmto message digestom sa bude neskôr porovnávať každé vloženie hesla. Výhodou tohto kódovania je to, že naše zadané heslo nefiguruje na serveri, namiesto neho je tam message digest, ktorý sa spätne nedá dešifrovať. Overenie užívateľa prebieha rovnako... zadané heslo sa pomocou hašovacej funkcie

premení na message digest a ten sa následne porovnáva s message digestom uloženým v systéme. Ak sa zhodujú užívateľ je autentifikovaný a je vpustený do systému. **Hašovacia funkcia je jednosmerná**, takže stráca údaje, ide tu len o overenie či došlo k zhode message digestov (V modených operačných systémoch sa na hašovanie používa hašovacia funkcia MD5 alebo SHA-1).

ELEKTRONICKÝ (DIGITÁLNY) PODPIS
-digitálny odtlačok správy vygenerovaný odosielateľom, sa zašifruje pomocou asymetrického algoritmu šifrovania, pričom odosielateľ digitálny odtlačok zašifruje (podpíše) svojím tajným kľúčom.
-čo sa týka funkčnosti, elektronický podpis je vlastne alternatívou tradičného ručného podpisu. Má za úlohu potvrdiť, že podpisujúci je skutočne ten, za ktorého sa vydáva a že súhlasí s obsahom podpísaného dokumentu.

Prednosťou elektronického podpisu je, že na rozdiel od klasického podpisu, ktorého overenie v prípade sporných situácií dokáže len znalec – grafológ (ale ani jeho výrok nemusí byť vždy jednoznačný a naisto správny), elektronický podpis je vždy jedinečný, keďže je istou jedinečnou kombináciou znakov (vytvorených definovaným matematickým postupom). Vďaka tomu výsledkom overovania elektronického podpisu je vždy výrok „áno“ alebo „nie“ so 100 % istotou.

Ďalšou odlišnosťou elektronického podpisu v porovnaní s klasickým je, že elektronický podpis (kombinácia znakov – núl a jednotiek) je závislý od obsahu dokumentu, ktorý podpisujete. To znamená, že ak by došlo k zmene obsahu podpísaného dokumentu (či už nejakou treťou osobou alebo poruchou pri prenose podpísaného dokumentu po internete), pri overení elektronického podpisu sa to hneď ukáže – odborne povedané EP zaručuje integritu správy (že pri prenose nedošlo k zmene obsahu správy). Okrem toho elektronickým podpisom nie je možné podpísať prázdny dokument („prázdny papier“) a naviac elektronický podpis je neprenosný na iný elektronický dokument (pretože závisí od obsahu podpísaného dokumentu). -autentifikácia. Ide tu o rozpoznanie a jednoznačnú identifikáciu osoby podpisujúcej určitý dokument. To znamená, že osoba, ktorej je dokument určený, má istotu, že odosielateľ je skutočne tá osoba, za ktorú sa vydáva.

-integrita. Integrita správy zaručuje to, že poslaný dokument sa dostal k adresátovi v nepozmenenej podobe, t.j. že to, čo odosielateľ odoslal, je skutočne to, čo prijímateľ dostal. -nepopierateľnosť. Nepopierateľnosť znemožňuje podpisujúcemu tvrdiť, že podpis nie je jeho a že to nebol on, kto daný dokument podpísal a odoslal. To znamená, že odosielateľ nemôže neskôr poprieť, že on poslal daný dokument. -šifrovanie

EP CERTIFIKÁT
-elektronický dokument podpísaný súkromným kľúčom certifikačnej autority.
-obsahuje verejný kľúč majiteľa certifikátu a ďalšie údaje týkajúce sa certifikátu ako aj držiteľa certifikátu – sériové číslo certifikátu, meno majiteľa, typ certifikátu, meno CA (Certifikačnej autority), elektronický podpis CA, dobu platnosti certifikátu a prípadne aj nejaké ďalšie údaje. -CA svojím podpisom na certifikáte potvrdzuje, že majiteľom verejného kľúča obsiahnutého v certifikáte je skutočne ten, kto je v popise certifikátu uvedený.

CERTIFIKAČNÁ AUTORITA (CA)
-inštitúcia, ktorá vystavuje certifikáty používané na identifikáciu majiteľa verejného šifrovacieho kľúča. S
-svojou funkciou sa podobá na štátneho notára.
-na Slovensku sú v súčasnosti 8 CA (najznámejšie sú PSCA, CA VÚB, D. Trust CA, CA Elektrotechnického výskumného a projektového ústavu (EVPÚ) – tieto 4 sú zároveň aj všetky Akreditované CA (ACA) na Slovensku (akreditované NBÚ))

ĎALŠIE ZÁKLADNÉ POJMY

- Autentifikácia – rozpoznanie a jednoznačná identifikácia osoby podpisujúcej určitý dokument. Potvrdenie, že odosielateľ je skutočne tá osoba, za ktorú sa vydáva.
- Časová pečiatka – potvrdenie, že nejaký dokument existoval v danom čase.
- Digitálny odtlačok – digitálny odtlačok je súhrn, zhrnutie správy. Digitálny odtlačok je pevnej dĺžky (nezávislej od dĺžky správy) a vypočíta sa pomocou tzv. hash funkcie.

Šifrovanie

Š

Obsah tohto článku

Šifrovanie jednotlivých správ

1. V správe na karte **Správa** v časti **Možnosti** kliknite na tlačidlo **Šifrovať obsah správy a prílohy**  .
2. Zostavte a pošlite správu.

Na začiatok stránky

Šifrovanie všetkých správ

1. V ponuke **Nástroje** kliknite na položku **Centrum dôveryhodnosti** a potom kliknite na položku **Zabezpečenie e-mailu**.
2. V časti **Zašifrovaný e-mail** začiarknite políčko **Zašifrovať obsah a prílohy odosielaných správ**.
3. Ak chcete zmeniť ďalšie nastavenia, napríklad zvoliť použitie určitého certifikátu, kliknite na tlačidlo **Nastavenia**.
4. Kliknite dvakrát na tlačidlo **OK**.

Poznámky:

- Ak chcete odosielať šifrované správy prostredníctvom Internetu, musíte si s príjemcom vymeniť certifikáty (certifikát: digitálne potvrdenie vašej identity. Keď odošlete digitálne podpísanú správu, posielate svoj certifikát a verejný kľúč.

Certifikáty vydáva certifikačná autorita a podobne ako vodičský preukaz, majú dátum exspirácie a môžu byť zrušené.) (súbory .cer (súbor .cer: súbor, ktorý obsahuje certifikát s verejným kľúčom, ale bez osobného kľúča. Súbory .cer importujte do priečinka Kontakty kliknutím na tlačidlo Import na karte Certifikáty a potom použite certifikát na odoslanie šifrovanej správy.)). Môžete tak urobiť niekoľkými spôsobmi. Napríklad:

- Odošlite digitálne podpísanú e-mailovú správu. Príjemca pridá vaše meno príjemcu e-mailových správ do svojho priečinka **Kontakty** a spolu s ním pridá aj váš certifikát.
- Odošlite e-mailovú správu s pripojeným súborom .cer alebo poskytnite príjemcovi disketu s daným súborom. Príjemca môže importovať súbor .cer do vašej vizitky kontaktu.
- Vytvorte vizitku kontaktu so svojím súborom .cer a odošlite ju.
- Publikujte svoj certifikát v adresári LDAP (Lightweight Directory Access Protocol (LDAP): protokol zabezpečujúci prístup k internetovým adresárom.) alebo v inom adresári, ku ktorému má daná osoba prístup.
- Zverejnite svoj certifikát na zdieľanom mieste, ku ktorému má daná osoba prístup.

- Ak správca systému nastavil zabezpečenie siete pomocou servera Microsoft Exchange, certifikáty sa nemusia vymieňať.
- Predvoleným šifrovacím algoritmom je 3DES (3DES: šifrovací algoritmus založený na štandarde DES (Data Encryption Standard). 3DES opakuje štandard DES trikrát. Z tohto dôvodu je 3DES pomalší ako štandard DES, je však bezpečnejší.). V USA už nie je stupeň šifrovania obmedzovaný vládou. Ak používate 40-bitový operačný systém, ktorý neumožňuje 128-bitové šifrovanie, program Outlook používa štandardne algoritmus RC2.