Semestrální práce

Autor: Jaroslav Körner

Kontrolní součty (MD5 a jiné) a jejich použití.

Kontrolní součty

Kontrolní součet pochází z anglického: **checksum = check + sum**, tedy spojení dvou slov *kontrola* a *suma*. Kontrolní součty slouží pro detekci chyby v datech. Dle webového IT slovníku, pak definice tohoto pojmu zní:

> **Checksum** (Kontrolní součet) - číslo, které slouží k kontrole integrity dat. Je vypočteno nějakým speciálním algoritmem. Kontrolní součet je přenášen společně s nějakou informací. Pokud vypočtený kontrolní součet nesouhlasí s tím přeneseným, nastala pravděpodobně při přenosu chyba.
>
> - IT slovník

Jak by ale takový kontrolní součet mohl vypadat?

Pro představu můžeme uvažovat velmi jednoduchý algoritmus, který bude na data aplikovat funkci $sum()\ mod(32)$ nad 32bitovými slovy. Když chceme odeslat zprávu po nespolehlivém kanálu, tak je potřeba ověřit na straně příjemce zda obdržel správná data. Odesilatel pak může postupovat následovně:

1. Rozdělí zprávu na $n * 32b$ úseků (poslední doplní nulami na příslušnou délku).
2. Začne tyto bloky postupně číst a odesílat kanálem. Zároveň je však bude sčítat "$sum()$" s akumulátorem (ten zde inicializujeme na hodnotu 0), pokud něco přeteče přes $32bitovou$ přesnost, tak se tyto bity jednoduše zahodí "$mod(32)$".
3. Když odesilatel odešle poslední slovo zprávy, tak přichází na řadu náš akumulátor. U něj se nejdříve určí jeho **dvojkový doplněk** (inverzní prvek pro sčítání), tím se docílí že příjemce po přičtení všech přijatých slov získá 0.
   Příjemce postupně jednotlivá slova k sobě sčítá obdobně jako odesilatel, ale mohou zde nastat dvě výsledné situace:
4. V akumulátoru mu zůstane po přijetí zprávy 0. Hurá! Zpráva je v pořádku.
5. V akumulátoru není 0. Něco se cestou pokazilo a my žádáme o znovu zaslání zprávy.

Takovýto algoritmus je velice triviální a praxi se moc nepoužívá. Vrátil by nám naprosto stejný výsledek například když by jsme proházeli jednotlivá slova zprávy. Ta tak nabyde naprosto jiného významu, ale její kontrolní součet bude naprosto shodný.

Příklady typů kontrolních součtů:

CRC - Cyclic Redundancy Check

Principem bývá zbytek po dělení polynomů na konečném tělese. V počítači jde o implementaci pomocí XOR nad binárními čísly, díky čemuž není implementace příliš složitá na výpočet.

Mezi rozšířené varianty CRC patří například:

- DNP
- CRC-8 Dallas
- CRC-16 XOR
- CRC-16 (IBM, Modbus)
- CRC-32
- CRC-8 Sum
- CRC-16 Sum
- CRC-32 Sum

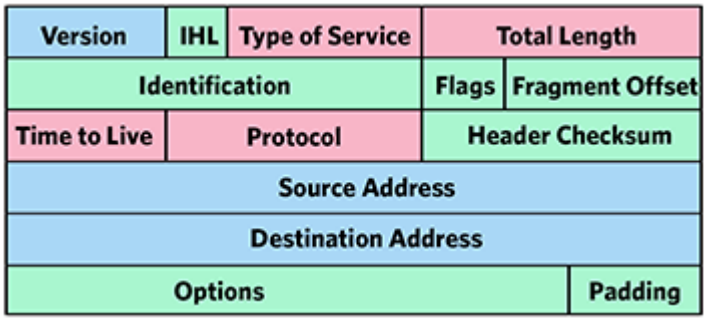
Varianty Sum se liší jen bitovostí výsledku, ale jejich výpočet stejný jako jsme si uváděli v postupu výpočtu triviálního kontolního součtu.

Použití kontrolních součtů:

Kontrolní součty se používají prakticky vždy když přenášíme nějaká digitální data. Jejich aplikaci můžeme najít například jmenovitě u technologií: **Bzip2**, **Ethernet** (IEEE 802.3), **Gzip**, **MPEG-2**, **PNG**, **SATA**, **Zip**, **Btrfs**, **Ext4**, **iSCSI**, **SCTP**...

IPv4 checksum

Internet protokol verze 4 také používá kontrolní součet, pro detekci chyby v hlavičce packetu. Používá se modifikovaného 16-SUM algoritmu.



ISBN-10

U ISBN-10 se kontrolní číslice získá tak, aby zbytek po dělení váženého součtu všech číslic jedenácti byl nulový. Kontrolní číslice tedy může mít i hodnotu 10, ta se zapisuje znakem X. Jako váhy se používají čísla (zleva doprava) 10 (pro první cifru kódu skupiny), 9, 8, …, 3, 2, 1 (pro kontrolní číslici).

Příklad kontroly platnosti ISBN-10:

	Skupina		**Vydavatel**			**Vydání**				**Kontrola**
ISBN	8	0	2	0	4	0	1	0	5	9
váhy	10	9	8	7	6	5	4	3	2	1
výsledek	80	+ 0	+ 16	+ 0	+ 24	+ 0	+ 4	+ 0	+ 10	+ 9

Výsledný součet je 143, kontrola: $143\ mod(11) = 0$ ($143 = 13 * 11$), ISBN 80-204-0105-9 je platné.

Čárové kódy EAN-13

EAN-13 se skládá z 13 číslic, kdy poslední z nich je kontrolní číslice. Ta je dopočítána pomocí funkce modulo 10.

Postup výpočtu (kód 8593026341407):

1. Sečteme číslice (od konce) na sudých pozicích $(4 + 4 + 6 + 0 + 9 + 8) = 31$
2. Přičteme součet číslic na lichých pozicích (od konce) vynásobený třemi $((0 + 1 + 3 + 2 + 3 + 5) * 3 = 42)$
3. Tento součet zaokrouhlíme na desítky nahoru $(31 + 42 = 73) \Rightarrow 80$
4. Kontrolní číslici získáme odečtením $80 - 73 = 7$

Stejným způsobem se kontrolní číslice vypočítává i pro **EAN/UCC8**, **EAN/UCC14** nebo pro číslo **SSCC**.

Paměti RAM s ECC

Zkratka ECC znamená (Error Correction Code), tedy paměti s podporou detekce chyb. Jejich využití je zejména v serverech a pracovních stanicích pro kritické fyzikální simulace. Jedna z implementací využívala kontrolu za pomoci redundantních bitů v paměti RAM, v podobně paritních bitů, které představovaly (sudou nebo lichou) paritu malého množství dat (obvykle jeden bajt) uložený v RAM, a následné porovnání uložené a vypočtené parity za účelem zjištění, zda nedošlo k chybě dat.

Hashovací funkce

Hashovací funkce nám dávají "otisk" souborů. Ten by měl být pro daný soubor stejně unikátní jako otisk prstu u člověka. Dobrá hashovací funkce musí splňovat do určité míry několik vlastností:

- nesmí být příliš rychlá na výpočet (ochrana proti brute-force),
- běžně v ní nenastávají kolize (kdy dva různé vzory mají stejný obraz),
- neexistuje k ní inverzní funkce (nelze z otisku určit původní soubor).

Dále také mývají tu vlastnosti že otisky dvou "podobných" souborů (například záměna jednoho písmene ve zprávě) si vůbec podobné nejsou.

Příklad zprávy kterou odesíláme nespolehlivým kanálem

Královna posílá zpráva po vládním poslovi:

1. "Jsem těhotná asi s kočím z Mostu. Královna"
2. "Jsem těhotná, asi skočím z mostu. Královna"
   MD5 otisky:
3. `3c723007c372b83ea3e28161f174dbdf`
4. `04005f28f6f6bed78a8a316df66a07f7`

Jak si můžete všimnout, stačí jen malá záměna zprávy vzniklá například tím že se královský posel přeslechne a král se dozví naprosto odlišnou informaci. Kdyby královna součástí zprávy poslovi předala i MD5 otisk, nemohli by již dojít k tak triviální záměně.

Při ověření souborů zaslaných po internetu neklademe zas takový důraz na to aby výpočet hashe byl časově náročný jako tomu je například u ukládání hesel.

Co je to ten Hash?

Jedná se o digitální otisk souboru, který se vygeneruje kryptografickou hashovací funkcí. Mezi příklady standardních hashovacích algoritmů patří:

- MD5,
- SHA1,
- RMD160,
- SHA256,
- SHA384,
- SHA512,
- SHA224.

Z nich jsou nejčastěji používány algoritmy SHA-1, SHA-256 a MD5, který však je dnes považován za "deprecated" tedy zastaralý a neměl by se používat pro nic důležitého. SHA je anglická zkratka slovního spojení: **Secure Hashing Algorithm**, tedy bezpečný hashovací algoritmus.

Postup práce s těmito otisky pak spočívá v porovnání kontrolního součtu získaného otisknutím staženého souboru s původním kontrolním součtem od jeho tvůrce. Ukázka kontrolního součtu získaného při použití kryptografické hashovací funkce SHA-256 může vypadat následovně:

`79caae92ee6a3ef8f639b31b6b516100449502c07df722ce5a68c407129a0f6d`

Tato metodika slouží jako kontrolní mechanismus při předávání souborů k ověření zda nedošlo k poškození předávaných dat, zda nedošlo k infiltrování dat škodlivým kódem a nebo zda nedošlo k jakémukoliv zásahu do dat během procesu předávání třetí stranou.

MD5 blíže

MD5 je akronym pro **Message-Digest algorithm 5**. Jedná se o rozšířený hashovací algoritmus. Ze vstupních dat vzniká výstup označovaný jako tzv. Hash nebo "otisk MD5". Při výpočtu se používá nasčítání hodnot do (4x32b) vnitřcích akumulátorů které jsou nastavené na předem definované hodnoty.

Nastínění postupu algoritmu MD5:

1. Inicializace akumulátorů.

```
word A: 01 23 45 67
word B: 89 ab cd ef
word C: fe dc ba 98
word D: 76 54 32 10
```

2. Vstupní data prohání funkcemi využívajících operace `XOR`.

```
F(X,Y,Z) = XY v not(X) Z
G(X,Y,Z) = XZ v Y not(Z)
H(X,Y,Z) = X xor Y xor Z
I(X,Y,Z) = Y xor (X v not(Z))
```

3. Pak se provede netriviální proházení bitů přes převodní tabulku.
4. Hodnoty se přičtou do akumulátorů:

```
A = A + AA
B = B + BB
C = C + CC
D = D + DD
```

5. Pokračuje se na kroku 2. dokud se nezpracuje celá zpráva.
6. Obsah akumulátorů se na výstupu zřetězí do výsledného 128b otisku.

Počítače mají dnes již dostatečný výkon na to, aby MD5 nesplňoval požadavek na "pomalý" výpočet, za účelem odolnosti vůči útokům typu hrubé síly (brute-force).

K čemu slouží hashování?

Slouží zejména k ověření autenticity souborů při internetové komunikaci. Poskytují ochrana jak proti chybě přenosu způsobené nespolehlivým médiem (kontrolní součet), tak i proti

úmyslné záměně souborů (otisk hashovací funkce).
Mezí další využití hashovacích funkcí patří ukládání hesel (kryptografické hashovací funkce), nebo například asociativní vyhledávání v datech. U vyhledávání klademe trochu jiné nároky na hashovací funkce. Například je výhodou, když data s podobnou vnitřní strukturou získají ve výsledku stejný nebo podobný hash, na čemž je pak postavené samotné vyhledávání či porovnávání souborů.

Zdroje:

- Rivest-MD5_rfc1321.pdf
- Kontrolní součty a jejich výpočty v C.pdf
- https://it-slovnik.cz
- Computerphile: Hashing Algorithms and Security
- Lukas Hron - kontrolni soucet checksum
- googlesource: crc32
- EAN13Barcode
- rfc: ISBN
- rfc: Computing the Internet Checksum