KIV/ZEP - 2011

Lehký úvod do komprese, checksumů a šifrování

Komprese dat

- redukce zabraného místa
- ztrátová vs. neztrátová komprese
  - RLE, Huffman, Waveletová transformace
- progresivní komprese
  - vhodné pro obrazová data případně zvuk
  - data dekomprimovány nejprve v nízké kvalitě a jak dekomprese postupuje, kvalita roste
  - není-li dostatek času dekomprimovat v nejvyšší kvalitě, uživatel alespoň vidí nízkou
  - některé části mohou zůstat v nízké kvalitě, protože nejsou ve středu pozornosti

Komprese dat

- komprese dat v paměti
  - redukuje problém s fragmentací paměti
  - umožňuje zpracování rozsáhlejších dat
  - příklady komprese
    - „undo“ informace
    - načtené pluginy (resp. jejich stavových dat)
    - volumetrická data (např. CT hlavy)
  - úsporné datové struktury vs. datové struktury se snadnou aktualizací
    - lomená čára – pole vs. spojový seznam vs. strom
    - Freemanova růžice

Checksums

- kontrola, že nějaká informace je správná
  - rodné číslo: musí být dělitelné 11
    - platí pro rodná čísla desetimístná (od roku 1954)
  - čísla kreditních/debitních karet (VISA, MASTERCARD, AMEX, ...), čísla sociálního pojištění v USA a Kanadě
    - LUHN algoritmus

Checksums

- LUHN algoritmus
  - postup:
    - sečíst číslice na liché (bráno od 1) pozici (zprava)
    - vynásobit každou číslici na liché pozici (počítáno od 1 zprava) jedničkou a na sudé pozici dvojkou
    - sečíst číslice čísel z předchozího kroku
    - je-li součet dělitelný 10, je číslo správně zadané
  - vlastnosti:
    - dokáže detekovat chybu na jednom místě
    - nedokáže detekovat dva zaměněné páry
    - nedokáže detekovat záměnu 22 za 55, 33 za 66 a 44 za 77

Checksums

- kontrola, že se nějaká data nezměnila
  - požaduje se vysoká přesnost zabezpečení - detekce poruchy přenosového média
  - CRC
    - bitová posloupnost = polynom *n*-tého stupně
    - CRC*k* = zbytek po dělení vstupního polynomu polynomem *k*-tého stupně (vhodně zvoleným)
    - CRC32: $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

Checksums

- CRC
  - postup:
    - máme dvě bitové posloupnosti: vstupních data a zvoleného polynomu – maska
    - maska propagována bit po bitu vstupními daty
    - je-li aktuální bit na vstupu = 1, změň vstup XOR operací mezi bity masky vstupní bitovou posloupností
    - nelze-li pokračovat, posledních *k* bitů je CRC*k*
  - vlastnosti:
    - velmi dobře se vypořádá s nahodilými poruchami
    - dvě zcela odlišné posloupnosti mohou mít totožné CRC
    - je poměrně snadné nalézt takovou posloupnost

Checksums

- MD5
  - 128 bitů, obvykle v hexadecimální soustavě, tj. např. „9e107d9d372bb6826bd81d3542a419d6„
  - výpočet složitější než u CRC
    - XOR, AND, OR, NOT operace
  - použitelné pro zabezpečení zprávy

Checksums

- kontrola, že se nějaká vstupní data nezměnila, takže není třeba provést jejich předzpracování, lze využít informace vypočtené v předchozím volání
  - musí být velmi rychlá
  - nemusí být přesná
    - analogie k hash funkcím
    - často stačí zkontrolovat jen část dat
  - Adler32

Checksums

- Adler32
  - postup:
    - A = 1, B = 0
    - dokud nejsou všechny BYTY zpracovány
      - A = A + aktuální byte MOD 65521
      - B = B + A MOD 65521
    - Adler32 = B << 16 | A
  - vlastnosti:
    - velice jednoduchá implementace
    - rychlý výpočet (lze jako hash funkce)
    - slabá ochrana, zejména pro málo dat

Checksums

- zabezpečení přístupu
  - je-li checksum zadaného hesla stejný jako uložené checksum, uživatel získá přístup
  - vygenerovat heslo odpovídající uloženému checksum je obtížné
    - MD5, SHA-1 (180 bitů), SHA-512 (512 bitů)
  - pozor na sociální inženýrství!

Základy kryptování

- zakryptujte zprávu: „DNES JE HEZKY“
  - pozpátku: „YKZEH EJ SEND“
  - posun o -1 písmeno: „CMDR ID GDYJX“
    - C = (P + b) mod m
    - P = (C - b) mod m
  - posun o -1 (lichá)/+1 (sudá): „CODT IF GFYLX“
  - nahrazení písmen: „AHOJ DO BORKU“
    - Enigma (viz „Hon na ponorku“)
    - poměrně bezpečné, ale vyžaduje zaslání klíče
    - lze prolomit, je-li zpráva dostatečně dlouhá, na základě známé frekvence písmen

Základy kryptování

- RSA (Rivest-Shamir-Adelman)
  - $C = P^e \bmod n$
  - $P = C^d \bmod n$
  - (e, n) je veřejný klíč (volně dostupný)
  - (d, n) je soukromý klíč (skrytý)
  - kdokoliv s (e, n) může zašifrovat zprávu pro vlastníka (d, n), ale nikdo kromě vlastníka ji neumí dešifrovat (a obráceně)

- RSA Algoritmus
  - vygeneruj dvě náhodná velká prvočísla p, q
    - velká = 150 cifer a více
  - $n = p \cdot q$, $f = (p - 1) \cdot (q - 1)$
  - zvol e = libovolné přirozené číslo ≤ f, takové, že největší společný dělitel NSD(e, f) = 1
  - spočítej d = číslo takové, že $e \cdot d - 1$ je dělitelné beze zbytku číslem f

Základy kryptování

- RSA prolomení
  - (e, n) = (59, 91)
  - stanov všechny prvočísla <= 91
    - 2, 3, 5, … (celkem m)
  - vynásob každou dvojici čísel (m·m) a nalezni takovou, že výsledek je 91, tj. 7·13
  - stanov f = 6·12 = 72 a ověř, že NSD(59,72)=1
  - spočítej d = 11 (11·59 – 1 / 72 = 9)
  - v praxi ale p, q mají 2048 cifer (binárních), tj. m = $O(2^{2048})$ → předpokládá se, že nelze prolomit na současných počítačích

Základy kryptování

Zkouška ☺

Feedback

Co dále?