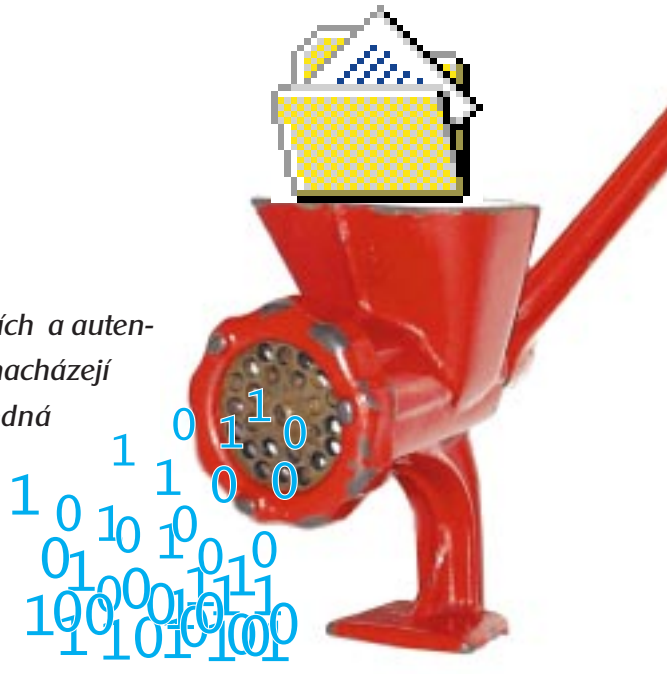
Hašovací funkce a kódy

Výživná haše



V komunikačních a bezpečnostních protokolech, přihlašovacích a autentizačních procedurách, certifikátech a digitálních podpisech nacházejí široké použití tzv. hašovací funkce a kódy. Protože se nejedná o triviální záležitost, vzniká mnoho pochyb a otázek. Na některé se nyní pokusíme odpovědět a popíšeme také současnou jedničku mezi hašovacími funkcemi, jíž je americký standard bezpečné hašovací funkce SHA-1.

Možná jste se už setkali s některou ze zkratek hašovacích funkcí a technik MD2, MD4, MD5, SNEFRU-128, SNEFRU-256, RIPEMD, RIPEMD-128, RIPEMD-160, SHA, SHA-1, TIGER, HMAC, MDx-MAC, RIPEMDx-MAC, N-HASH, FFT-HASH, CELLHASH, BOOGNISH, SL2 ap. Znalost hašovacích funkcí a technik se hodí programátorům, administrátorům sítí a každému, kdo se pohybuje kolem komunikací, zabezpečení dat, protokolů, autentizace atd. Tento velmi užitečný nástroj jim může pomoci řešit řadu úloh, na které ve své praxi narážejí.

K čemu to je

Příkladů použití hašovacích funkcí je nepřeberně, pro představu uveďme jeden triviální. Máme zkontrolovat, zda obsahy dvou databází o rozsahu několika TB uložených na dvou geograficky vzdálených počítačích jsou shodné. Kupodivu kvůli tomu nemusíme přenášet ony miliardy bajtů a porovnávat přímo jednotlivé datové bloky. Postačí, když vypočítáme hašovací hodnoty obou databází a tyto hodnoty porovnáme. Jejich délka je obvykle 128 nebo 160 bitů a jsou-li shodné, jsou téměř jistě (s pravděpodobností $1 - 1/2^{128}$, resp. $1 - 1/2^{160}$) shodné i příslušné databáze.

Pokud jste se s hašovacími funkcemi ještě nesetkali, můžete to nyní dohnat. Je načase, neboť např. při instalacích některého softwaru se po nás stále častěji chce, abychom si vybrali některou z nabízených hašovacích funkcí. Internetové prohlížeče a aplikace, síťový software, elektronická pošta, autentizace, digitální podpisy, bezpečnostní protokoly (např. SSL) ap. jsou jen některé příklady, kde se hašovací funkce používají. A budete-li něco digitálně podepisovat, ať už vědomě či nevědomě (moderní software to i bez vašeho vědomí za vás dělá dost často), vězte, že vlastně nepodepisujete žádnou zprávu, dokument, program nebo digitální identitu, ale právě jen hašovací hodnotu tohoto objektu (datového souboru).

Co to je

Vstupem ***hašovací funkce*** H je datový soubor M (zpráva) o proměnné a prakticky neomezené délce, jejím výstupem je ***hašovací hodnota*** (***hašovací kód***) H(M) pevné a relativně velmi malé délky (většinou jen desítky až stovky bitů).

Hašovací funkce plní dvě role. První rolí je „komprese“. Velmi dlouhý vstup (u hašovací funkce SHA-1 až $2^{64} - 1$ bitů!) je „komprimován“ na velmi krátký výstup (u SHA-1 je to 160 bitů). O komprimování mluvíme v uvozovkách proto, že původní informace nemůže být v hašovacím kódu obsažena celá, takže z hašovacího kódu není možné obnovit původní zprávu. Jak se této vlastnosti dá skvěle využít, uvidíme dále.

Druhou rolí je vlastní ***hašování***, tedy „semletí“ vstupních dat (výstupní data jsou skutečně jakýmsi vzorkem „semletých“ vstupních dat). Mlýnkem zde sice jsou matematické funkce, ale jinak je podobnost s reálným mletím velmi příhodná – ostatně původní anglické slovo *hash* znamená mleté maso.

Hašovací hodnota je přitom jen výtažkem z takto získaného produktu (zůstaneme-li u kuchyňské analogie: z veškerého namletého masa jsme si pro ochutnání odebrali trochu na lžíci, což stačí pro posouzení celku). V případě zprávy se nazývá *message digest*, tj. doslova výtah zprávy, u člověka by hašovací hodnotě mohl odpovídat třeba otisk prstu nebo genetický kód ap. A také zde si povšimněte jedné důležité vlastnosti: z otisku prstu nelze člověka vytvořit, ale lze ho podle něj jednoznačně identifikovat.

Pojem ***hašovací techniky*** se objevil později. Bývá používán zpravidla pro označení obecnějšího postupu zpracování dat, v němž je hašovací funkce použita jako jeden stavební blok. Volbou konkrétní hašovací funkce pak definujeme konkrétní instanci hašovací techniky.

Bezpečnostní požadavky

Aby hašovací funkce správně pracovaly, musí splňovat následující požadavky:

Jednocestnost:

- je-li dáno M, je jednoduché vypočítat H(M);
- je-li dáno H(M), je velmi těžké (rozuměj: výpočetními prostředky prakticky neproveditelné) vypočítat M;
- je-li dáno M, je velmi těžké nalézt M´ tak, aby H(M) = H(M´).

Odolnost proti kolizi:

- je velmi těžké nalézt jakékoliv (tedy i náhodné) různé M a M´ tak, aby H(M) = H(M´), tj. aby došlo k tzv. ***kolizi***.

Pokud si tyto vlastnosti z modelu „zpráva – hašovací kód“ přeneseme do situace „člověk – otisk prstu“ dostaneme tyto bezpečnostní požadavky ve velmi srozumitelné formě:

Jednocestnost:

- otisk prstu lze pořídit snadno;
- máme-li otisk prstu, je těžké (prakticky nemožné) z něj rekonstruovat jeho nositele;
- máme-li k dispozici nějakého konkrétního člověka, je těžké k němu najít jiného se stejným otiskem prstu.

Odolnost proti kolizi:

- je velmi těžké nalézt jakékoliv dva lidi, kteří by měli stejný otisk prstu.

Báječná jednocestnost

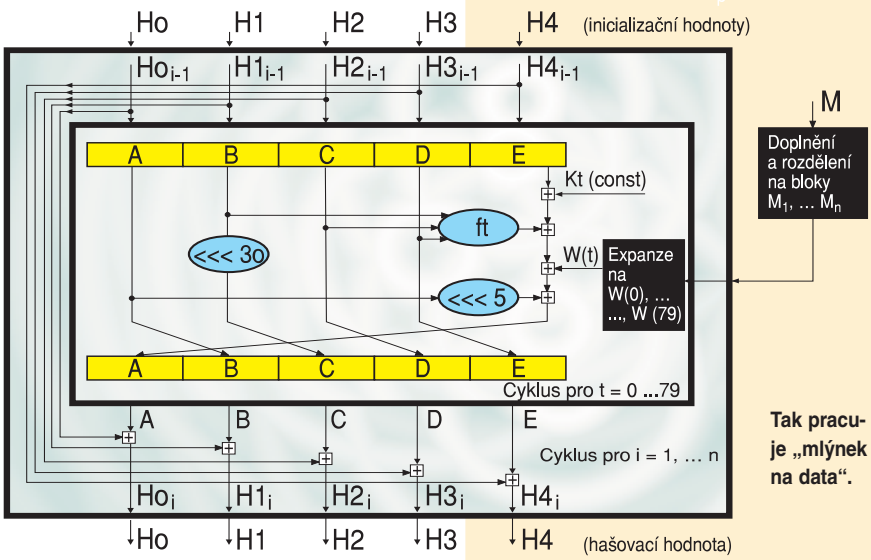
Další klasickou aplikací hašovacích funkcí je ukládání hesel nebo šifrovacích klíčů. Právě zde se využívá jednocestnosti. Když se s přihlašovacími hesly (passwordy) začínalo, v systému se ukládala hesla všech jeho oprávněných uživatelů v otevřeném tvaru. Později se místo hesel (PSW) začaly ukládat jejich hašovací hodnoty H(PSW), z nichž díky jednocestnosti už nešlo vypočítat PSW. Prostě řečeno, hašovací hodnota umožňuje identifikovat, kontrolovat, ale nikoli rekonstruovat.

V případě šifrovacích klíčů je situace obdobná. Představme si zašifrovaný soubor, který je našemu softwaru předán k dešifrování. Nejprve jsme vyzváni k zadání šifrovacího klíče. Poté systém tímto klíčem soubor rozšifruje a na místo původního souboru uloží jeho odšifrovaný tvar. Pak ale třeba zjistíme, že soubor je odšifrovaný špatně, protože jsme použili jiný, nesprávný klíč. To by se pochopitelně nemělo stát – proto se v hlavičce zašifrovaného souboru zpravidla uvádí nějaká informace o klíči, například je možné umístit zde právě hašovací hodnotu správného klíče. Po zadání klíče se tedy nejprve vypočítá jeho hašovací kód a výsledek se porovná s hodnotou uloženou v hlavičce zašifrovaného souboru. Správný klíč tak lze pomocí hašovací hodnoty identifikovat, aniž by ho z ní byl kdokoli schopen zrekonstruovat.

Pozor na kolize!

Proč musí být hašovací funkce odolné proti kolizím, nejlépe demonstruje příklad hašovací funkce COMP128. Psali jsme o ní v seriálu o bezpečnosti telefonů GSM (viz Chip 7/98, str. 134). Byla to právě nedostatečná odolnost proti kolizím, která vedla až k odhalení tajného klíče SIM karty a tudíž k možnosti neoprávněného klonování cizích SIM karet pomocí kryptoanalýzy.

Hašovací funkce SHA-1



Dalším příkladem zneužití kolize hašovací funkce může být digitální podpis. Digitální podpis zprávy M se až na výjimky pořizuje tak, že se digitálně podepisuje pouze výtah zprávy, tj. nejprve se vypočte hašovací hodnota H(M), a na tu se teprve aplikuje digitální podpis. Na tomto postupu a kolizích hašovací funkce H je založena obecná metoda, která se docela poeticky nazývá *How to swindle Rabin* podle stejnojmenného článku G. Yuvala z časopisu Cryptologia z roku 1979 (Vol. 3, No. 3, str. 187).

Jak podvést rabína

Zmíněná metoda se používá v různých modifikacích. Jedna z nich byla nazvána *How to swindle Ann* a Hans Dobbertin ji prezentoval na konferenci Fast Software Encryption v roce 1996. Představme si, že jistý Alf Blowfish chce podvést paní Annu Bonideovou, která má zájem o jeho dům. Dohodnou se na částce 176 495 dolarů. Alf pak sepíše smlouvu o prodeji domu za smluvenou částku a požádá Annu o její digitální podpis. Protože ví, že Anna použila k digitálnímu podpisu hašovací funkci MD4, u níž je schopen konstruovat kolize, dohodu sepsal tak, že má k dispozici zcela jinou smlouvu (viz druhý text v rámečku), která je pro něj o 100 000 dolarů výhodnější a má stejnou hašovací hodnotu jako smlouva podepsaná Annou. Tím má v ruce digitálně podepsaný dokument, který Anna nepodepsala a je pro ni nevýhodný.

****
CONTRACT

At the price of $176,495 Alf Blowfish sells his house to Ann Bonidea. .....

****
CONTRACT

At the price of $276,495 Alf Blowfish sells his house to Ann Bonidea. .....

Na uvedeném příkladu jsou důležité dvě okolnosti:

- jde o zcela konkrétní text, který skutečně vede ke kolizi u hašovací funkce MD4;
- prezentovaný postup by se dal s mírnou modifikací použít i u elektronických platebních protokolů využívajících funkci MD4.

SHA-1 je bezpečná

Přejděme nyní k popisu funkce, která má všechny požadované bezpečnostní vlastnosti zabraňující jejímu zneužití. ***SHA-1*** byla vytvořena americkou tajnou službou NSA a americkým úřadem pro normalizaci NIST byla vyhlášena 17. 4. 1995 jako standard v oficiálním dokumentu Federal Information Processing Standards Publication 180-1 (FIPS PUB 180-1). Je určena nejen pro potřeby algoritmu digitálního podpisu (DSA), ale i pro všechny aplikace ve státním sektoru, kde je požadována bezpečná hašovací funkce. SHA-1 tak nahradila svoji předchůdkyni SHA, definovanou v FIPS PUB 180 z 11. 5. 1993. Dokumenty jsou nazvány *Secure Hash Standard* (SHS), přičemž vlastní algoritmus se nazývá *Secure Hash Algorithm* (SHA). Rozdíl mezi definicí SHA-1 a SHA je nepatrný: SHA-1 má v příkazovém řádku (b) hlavní smyčky (viz dále) jednu jednobitovou rotaci navíc, ale rozdíl mezi jejich bezpečností je velký. Funkce SHA-1 je považována za bezpečnou, zatímco SHA nikoli.

Popis SHA-1

SHA-1 byla navržena jako standardní hašovací funkce se vstupem od 0 až do 2^{64} – 1 bitů a výstupem 160 bitů. Myšlenkově vychází z návrhu algoritmu MD4 od R. Rivesta (1990),

ale velmi posiluje jeho vnitřní funkce, takže zatímco u MD4 již byly nalezeny kolize, SHA-1 je vůči nim považována za rezistentní. Její algoritmus sestává z několika hlavních kroků, které si v dalších odstavcích popíšeme podrobněji.

Doplnění zprávy

Předpokládejme, že máme zprávu M, která má **m** bitů. SHA-1 zpracovává bloky dat po 512 bitech, takže nejprve dojde k doplnění zprávy M na délku, která je celočíselným násobkem 512 bitů. Doplnění se dělá v každém případě a je definováno tak, že M se nejprve doplní jedničkovým bitem a poté 0 až 511 nulovými bity tak, aby délka zprávy byla rovna 512*(**n** – 1) + 448, kde **n** je vhodné přirozené číslo. Zbývajících 64 bitů bude doplněno 64bitovým číslem, které vyjadřuje počet bitů původní zprávy (**m**). Poznamenejme, že prázdná zpráva má **m** = 0 bitů a doplňuje se stejným způsobem jako každá jiná. Takto vždy vznikne **n** 512bitových bloků dat, které označíme M1 až Mn (doplňování i popis se provádí v orientaci BIG ENDIAN).

Logické funkce a konstanty

V dalším popisu budeme pracovat s 32bitovými slovy (dále jen slova) A až E a TEMP, konstantami H0 až H4, K0 až K79 a funkcemi f0 až f79.

Vstupem každé funkce ft, kde $0 \le t \le 79$, jsou tři slova B, C, D. Výstupem ft je slovo definované takto:

ft(B,C,D) = (B AND C) OR
((NOT B) AND D) pro $0 \le t \le 19$,
ft(B,C,D) = B XOR C XOR D pro $20 \le t \le 39$,
ft(B,C,D) = (B AND C) OR (B AND D)
OR (C AND D) pro $40 \le t \le 59$,
ft(B,C,D) = B XOR C XOR D pro $60 \le t \le 79$.

Dále se využívají konstanty K0 až K79, které jsou v hexadecimálním zápisu rovny:

Kt = 5A827999 pro $0 \le t \le 19$,
Kt = 6ED9EBA1 pro $20 \le t \le 39$,
Kt = 8F1BBCDC pro $40 \le t \le 59$,
Kt = CA62C1D6 pro $60 \le t \le 79$.

Před zpracováním bloku M1 se H0 až H4 nastaví na inicializační hodnoty:

H0 = 67452301,
H1 = EFCDAB89,
H2 = 98BADCFE,
H3 = 10325476,
H4 = C3D2E1F0.

Dále označme „W <<< s“ cyklickou rotaci slova W o **s** bitů doleva. Výpočet hašovací hodnoty se provádí postupným zpracováním bloků M1 až Mn, jak je uvedeno dále.

Hlavní smyčka

Zpracování bloku Mi (i = 1 až **n**) probíhá v pěti krocích (a) až (e):

(a) Mi rozdělíme na 16 slov W(0) až W(15).

(b) Provedeme expanzi na slova W(16) až W(79):

W(t) = (W(t-3) XOR W(t-8) XOR W(t-14) XOR W(t-16)) <<< 1.

(c) Do A až E zkopírujeme poslední hodnoty slov H0 až H4:

A = H0, B = H1, C = H2, D = H3, E = H4.

(d) V následujících 80 rundách (t = 0, ... 79) ke slovům A až E přimícháváme slova W podle symbolického zápisu:

TEMP = (A <<< 5) + ft(B,C,D) + E + W(t) + + Kt
E = D
D = C
C = B <<< 30
B = A
A = TEMP

(e) Aktualizujeme hodnoty H0 až H4 podle vztahů:

H0 = H0 + A, H1 = H1 + B, H2 = H2 + C, H3 = H3 + D, H4 = H4 + E.

Po zpracování posledního bloku Mn je hašovací hodnota definována jako 160bitový řetězec tvořený slovy H0 až H4.

Závěr

Ukázali jsme, že hašovací funkce mají široké použití ve výpočetní praxi. Musí však splňovat náročná bezpečnostní kritéria, kterým však dnes vyhoví jen několik hašovacích funkcí. Patří mezi ně i SHA-1, důvěryhodný a velmi rozšířený americký státní standard, který se stal také světovým průmyslovým standardem.

Hašovací funkce a techniky jsou velmi rozšířené, takže bude jistě dobré mít určité znalosti o jejich vlastnostech a bezpečnosti. Příště se proto budeme věnovat dalším hašovacím funkcím a technikám a jejich srovnání z hlediska bezpečnosti a rychlosti.

Vlastimil Klíma (*vklima@decros.cz*)