Hašovací funkce a kódy

Jak se melou data

V tomto článku, který těsně navazuje na „Výživnou haši" z minulého Chipu, si všimneme dalších hašovacích funkcí a odpovíme na otázku, které z nich jsou bezpečné. Možná budete překvapeni, že značně rozšířené MD4 a MD5 se už nedoporučují a jejich místo zaujaly SHA-1 a RIPEMD-160.

Hašovací funkce jsou základem digitálních podpisů, certifikátů a bezpečnostních protokolů. Povědomí o nich nám proto dnes může být často užitečné. Ostatně v předchozím článku přišla paní Bonideová vinou nebezpečné MD4 v digitálním podpisu o 100 000 dolarů, nekvalitní hašovací funkce COMP128 v SIM kartách telefonů GSM umožňují jejich klonování atd. – opatrnost je tedy určitě na místě.

Tři nejznámější

Hašovacích funkcí existují desítky. Nejrozšířenější, s kterými se určitě setkáte, jsou tři hlavní třídy: MDx, RIPEMD-x, SHA-x, kde x označuje příslušnou verzi. Tyto funkce z velmi dlouhé zprávy M (soubor dat o délce až 2^{64} bitů) vytvoří hašovací kód o délce 128, resp. 160 bitů. Kompresi uvedených hašovacích funkcí zajišťuje tzv. kompresní funkce (f). U zmíněných funkcí je zpráva M před vlastním hašováním doplněna a zarovnána na celistvý počet 512 bitových bloků Mi, i = 1..n, a dále je definována inicializační hodnota IV (konstanta příslušné hašovací funkce). Proces hašování využívá kompresní funkci iterativně takto:

$$H_0 = IV,$$
$$H_i = f(H_{i-1}, M_i),\ i = 1..n,$$
$$H(M) = H_n.$$

Funkce f je u každé hašovací funkce definována jinak, ale toto schéma je platné pro většinu z nich. Například pro SHA-1 je kompresní funkce f popsána v předchozím článku v tzv. hlavní smyčce.

Jakou délku hašovacího kódu?

Zprávy M mohou mít až 2^{64} bitů, hašovací kódy mají jen desítky bitů, mnoho zpráv tedy musí mít shodné kódy. Například pro 10bitový hašovací kód postačí vygenerovat 1025 zpráv, abychom měli jistotu, že dvě z nich mají stejný hašovací kód. Kolize znamená, že H(M) = H(M´) pro různé M a M´. V praxi, díky tzv. narozeninovému paradoxu (blíže viz např. Chip 7/98, str. 136), postačí nikoli 2^d, ale pouze $2^{d/2}$ zpráv, aby ke kolizi došlo s pravděpodobností cca 50 %. Pro délku kódu d = 256 bitů je to 2^{128} zpráv a ty už nejsme schopni vygenerovat. ***Bezpečné hašovací funkce*** neumožňují nalezení kolizí žádným účinnějším postupem a délka kódu má zaručit, aby hledání kolizí narozeninovým paradoxem ($2^{d/2}$ zpráv) bylo výpočetně nezvládnutelné. První hašovací funkce používaly délku kódu 128, nyní je to 160 bitů a více.

Nejznámější hašovací funkce už nedoporučuje používat ani jejich autor.

160 bitů odolá dlouho

V roce 1994 byl P. Oorschotem a M. Wienerem navržen stroj v ceně 10 milionů USD, který je schopen vygenerovat 2^{64} kódů, a tudíž realizuje narozeninový paradox u 128bitového kódu (připomeňme existující DES Cracker, provádějící 2^{56} šifrování během 9 dní). Naproti tomu 160bitové hašovací kódy budou prakticky odolné minimálně v následujících cca 20 až 25 letech. Jednoduchým výpočtem zjistíme, že příslušný kolizní stroj by byl 2^{16}krát dražší, neboť by musel provést 2^{80} operací. Aby se dostal na cenu 10 milionů USD, musela by se technologie 2^{16}krát zrychlit. To podle známého Moorova zákona (bude-li platit i nadále) nastane za 24 let, což je velmi přijatelná doba pro platnost standardu. Pokud dojde ke změně standardu, 256bitové kódy by mohly zůstat konečnou délkou, protože lidé asi nebudou nikdy ochotni zaplatit za 2^{128} operací na zjišťování kolizí hašovacích funkcí.

Není kolize jako kolize

Je zřejmé, že jakmile se najde kolize u *hašovací* funkce, jak jsme to viděli u MD4 v minulém článku, je s takovou hašovací funkcí konec. O něco slabším útokem je nalezení kolize pro příslušnou *kompresní* funkci. Jde o nalezení různých M_i a M_i' a vhodného H_{i-1}, pro něž je $f(H_{i-1}, M_i) = f(H_{i-1}, M_i')$. I když nejde o použitelný výsledek, považuje se to za vážný bezpečnostní nedostatek – to je případ MD2 a MD5.

Stop hašovacím funkcím MD

Autorem hašovacích funkcí MD je R. Rivest, zakladatel RSA Data Security Inc. (dnes divize Security Dynamics). Jako první vznikla MD2 (1989), která je bajtově orientovaná, pomalá a zcela jiná než její 32bitově orientované následnice MD4 (1990) a MD5 (1991). MD2 byla zapomenuta, MD4 zapovězena z bezpečnostního hlediska (kolize) a také funkci MD5 sám její autor R. Rivest nedoporučuje používat pro digitální podpisy a všude tam, kde by se mohlo využít kolizí. I kdybychom přehlédli varování autora MD5 a kolizi její kompresní funkce, zůstává námitka proti 128bitovému kódu.

Jinými slovy – z třídy funkcí MD už nezbylo vůbec nic (i příznivci známého programu PGP si jistě všimli, že místo MD5 se začala používat SHA-1). Funkce MD „poslal ke dnu“ pracovník německé bezpečnostní informační služby (GISA) H. Dobbertin – na podzim 1995 MD4 a na jaře 1996 MD5. Popis MD2, 4 a 5 naleznete jako RFC 1319-21, jejich použití pro komerční účely podléhá licenci.

Evropské hašovací funkce RIPEMD-x

Funkce ***RIPEMD*** je první z třídy RIPEMD-x. Byla navržena v rámci projektu RACE Integrity Primitives Evaluation (RIPE) Komise Evropských společenství, který měl pomoci evropské standardizaci kryptografických funkcí. V rámci projektu (završen v polovině 90. let) byly hodnoceny a navrženy různé kryptografické nástroje. RIPEMD vychází z MD4, ale je bezpečnostně posílena. Zajímavé je rozdělení kompresní funkce na dvě a kombinace jejich výsledků v závěru zpracování každého bloku. Kolize u ní nebyly nalezeny (jen v její zeslabené variantě), ale nevýhodou je 128bitový kód.

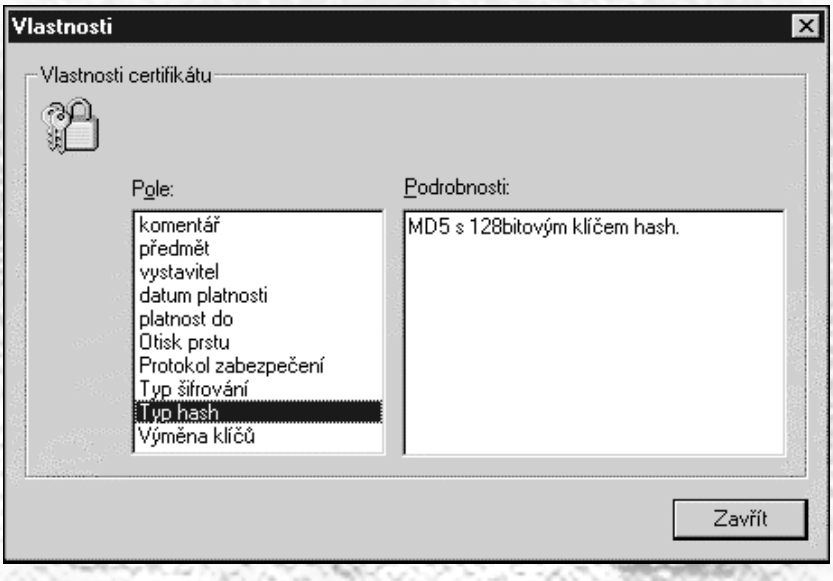
Proto v roce 1996 H. Dobbertin a dva Belgičané A. Bosselaers a B. Preenel (už mimo projekt RIPE) navrhli ***RIPEMD-160*** se 160bitovým hašovacím kódem. Zesiluje původní RIPEMD a výsledkem je skvělý návrh hašovací funkce (podrobnosti viz Infotipy). Navrhli také variantu RIPEMD-128 se 128bitovým kódem jako náhražku RIPEMD tam, kde nelze použít kód 160bitový.

Pro ty, kdo vyžadují ještě vyšší bezpečnost, byly vytvořeny dokonce i RIPEMD-256 a RIPEMD-320. Vzniknou vytvořením dvou paralelních linií zpracování dat pomocí kompresních funkcí RIPEMD-128 a RIPEMD-160, v nichž jsou navíc vzájemně kombinovány jejich vnitřní stavy. Vše o nich můžete zjistit na jejich domovské stránce *http://www.esat.kuleuven.ac.be/~bosselae/ripemd160.html*.

RIPEMD-160 je nejvážnějším dnešním protikandidátem SHA-1 a byla začleněna do mezinárodního standardu ISO/IEC 10118-3, společně s RIPEMD-128 a SHA-1. RIPEMD-128, 160, 256 a 320 jsou zaregistrovány jako funkce společnosti TeleTrusT, ale patří do freewaru a mohou se bezplatně použít i pro komerční účely.

Další hašovací funkce

Z mnoha a mnoha dalších zmíníme už jen dvě. ***HAVAL*** používá velmi dobré kryptografické nástroje a je dosud považována za bezpečnou. Navrhli ji v roce 1992 tři Australané na konferenci Auscrypt´92, má nastavitelnou vnitřní složitost i délku výstupního kódu (od 128 do 256 bitů po 32 bitech). Při nejvyšší nastavené složitosti je stejně rychlá jako MD5.



Hašovací funkce se uplatní například v certifikátech.

Americké SHA-0 a SHA-1

Funkci ***SHA-1*** se 160bitovým kódem jsme popsali v minulém čísle. Vznikla jako vylepšení SHA (pro odlišení označovaná SHA-0) a napravila jistý nedostatek, který SHA-0 měla. NSA ho zjistila v roce 1994, ale opravu nijak nekomentovala. O co šlo, poodhalili v srpnu 1998 Francouzi F. Chabaud a A. Joux, když nalezli útok na kompresní funkci SHA-0 rychlejší (se složitostí 2^{61}) než narozeninový paradox (složitost 2^{80}). I když je to hodně teoretický výsledek a týká se jen kompresní funkce, není důvodu, proč nepoužít bezpečnější SHA-1, vůči tomuto útoku odolnou.

TIGER je hašovací funkce navržená v roce 1996 pro 64bitové procesory (z tohoto důvodu není právě v centru pozornosti). Navrhli ji známí kryptologové Biham a Anderson, má 192bitový hašovací kód a rychlost hašování je cca 34 Mb/s. Více viz Infotipy.

Hašovací techniky

Mezi nejznámější hašovací techniky patří tzv. ***klíčované*** (kryptografické) hašovací funkce. Vystupují pod různými názvy, ale mají jedno společné: s daty „semelou“ do výstupního kódu také tajný klíč. Příkladem může být ***IBC-hash***, vytvořená v rámci RIPE. Hašovací kód detekuje jakoukoliv úmyslnou i neúmyslnou změnu ve zprávě, čímž zajišťuje integritu zprávy, současně ale také autentizuje jejího původce, neboť musel znát tajný klíč. Proto se tento hašovací kód nazývá ***autentizační kód zprávy*** (MAC, Message Authentication Code). Před vznikem hašovacích funkcí se k výpočtu MAC používaly blokové šifry. Psali jsme o nich v článcích „Nepaděla-

hašovací funkce	MD2	MD4	MD5	RIPEMD	RIPEMD--128	RIPEMD--160	SHA-0	SHA-1
hašovací kód	128 bitů	128 bitů	128 bitů	128 bitů	128 bitů	160 bitů	160 bitů	160 bitů
poznámka				2 paralelní linie	2 paralelní linie	2 paralelní linie		
bezpečnost	kolize kompresní funkce,1995, malá délka kódu	kolize celé MD4, 1995, malá délka kódu	kolize kompresní funkce, 1996, malá délka kódu	malá délka kódu	malá délka kódu	**bezpečná**	kolize kompresní funkce (ale vysoká složitost nalezení – 2^{61})	**bezpečná**
tvůrce a licenční politika	R. Rivest, RSA, licenční poplatek			TeleTrusT, freeware			vytvořila NSA, vydal NIST jako standard USA	
vznik	1989	1990	1991	1992 – 5	1996	1996	1993	1995
dokument	RFC 1319	RFC 1320	RFC 1321	link viz infotipy	link viz infotipy	link viz infotipy	NIST FIPS PUB 180	NIST FIPS PUB 180-1
orientační rychlost [Mb/s] v C (Pentium, 90 MHz)		81	60		36	19	21	21
orientační rychlost [Mb/s] v ASM (Pentium, 90 MHz)	4	191	136	96	78	45	55	55

Vlastnosti hašovacích funkcí.

telné zabezpečení dat“ (Chip 8/93, str. 166, 9/93, str. 212). Technika MAC byla standardizována jako norma ISO (viz ISO/IEC 9797) a mezi tyto funkce patří i ***RIPE-MAC1*** a ***RIPE-MAC3,*** vytvořené v rámci RIPE na základě blokových šifer DES a tripleDES.

Klíčované hašovací funkce MDx-MAC

Ilustrativním příkladem tvorby autentizačního kódu zprávy M s použitím hašovací funkce H a klíče K je například konstrukce G. Tsudika (1992):
MAC(M) = H(K || M) nebo H(M || K) aj., kde || označuje zřetězení dat. Se zprávou se tedy jako prefix nebo suffix „semele“ i tajný klíč.
V roce 1995 B. Preenel a P. C. Oorschot upozornili na možná úskalí takto jednoduše definovaných MAC a navrhli rodinu autentizačních kódů ***MDx-MAC***. Používají hašovací funkce „typu MDx“ (to jsou všechny hašovací funkce rodin MD, RIPEMD a SHA) jako základ, ale zesložiťují je. Jejich programovou realizaci s kontrolními příklady můžete nalézt na *www.esat.kuleuven.ac.be*, viz Infotipy. Zesložitění spočívá ve vstupu klíče do inicializační hodnoty IV i konstant kompresní funkce a v doplnění nového datového bloku (závislého na klíči) na konec zprávy.

Klíčované hašovací funkce HMAC

Tato technika byla navržena v roce 1997 a tvoří RFC 2104. Jedná se o postup vytvoření MAC pomocí libovolné hašovací funkce H a tajného klíče K. Konkrétně vzniklý MAC má pak název ve tvaru „HMAC-H“. Kontrolní příklady pro kódy ***HMAC-SHA1*** a ***HMAC-MD5*** tvoří RFC 2202. Kombinace klíče K s hašovací funkcí H a zprávou M probíhá tak, že se nejprve klíč doplní nulovými bity na délku bloku B bajtů (zde B = 64) a definují se konstantní B-bajtové bloky *ipad* a *opad*. Výsledný kód je pak vypočítán jako H((K xor opad) || H(K xor ipad) || M)), kde H je konkrétní instance hašovací funkce.

Závěr

Hašovací funkce prodělaly stejný vývoj jako blokové šifry a jejich nejnovější verze mají dostatečnou odolnost a důvěru. Patří mezi ně hlavně RIPEMD-160 a SHA-1. Z hašovacích technik známe osvědčenou tvorbu MAC na bázi kvalitní blokové šifry a za bezpečné jsou také považovány hašovací techniky MDx-MAC nebo HMAC v kombinaci s kvalitní hašovací funkcí.
V tomto dvoudílném článku jsme se pokusili vybrat alespoň to nejdůležitější z oblasti hašování. Máte-li hlubší zájem, stačí, abyste spustili svůj oblíbený internetový prohlížeč a napsali magické slůvko „hash“.

Vlastimil Klíma
(vklima@decros.cz)

infotipy

Definice standardu SHA-1:
http://www.itl.nist.gov/div897/pubs/fip180-1.htm
Rychlostní charakteristiky hašovacích funkcí a vše ke třídě RIPEMD-x:
http://www.esat.kuleuven.ac.be/~bosselae/
Vše o hašovací funkci Tiger:
http://www.cs.technion.ac.il/~biham/
Poučný článek popisující nalezení kolizí u MD4:
http://www.cs.ucsd.edu/users/bsy/dobbertin.ps
Všechny zmíněné dokumenty RFC:
http://info.internet.isi.edu/in-notes/rfc/files/