Hašovací funkce a kódy

Jednoznačné otisky dat

S příchodem nového šifrovacího standardu AES, jehož schvalování právě probíhá, se budou inovovat i další bezpečnostní techniky. Zavádějí se nové hašovací funkce SHA-256, SHA-384 a SHA-512, u nichž lze očekávat široké uplatnění, neboť nabízejí vyšší bezpečnost a zřejmě budou doprovázet nový AES i nová schémata elektronického podpisu.

S ***hašovacími funkcemi*** (rodinami MD, SHA, RIPEMD) a technikami (HMAC) jsme vás podrobně seznámili v číslech 3/99 a 4/99 (v elektronické podobě je naleznete na internetu, viz infotipy). Tyto funkce jsou velmi užitečné, a proto se s nimi budeme velmi často setkávat. Jsou používány pro elektronické podpisy, neboť vytvářejí „otisky" zpráv (a slouží nám podobně jako otisky prstů v daktyloskopii), a dále například pro kontrolu integrity dat, pro autentizaci v bezpečnostních protokolech, ke kontrole hesel a šifrovacích klíčů, pro techniky HMAC, v generátorech náhodných bitů apod.

JEDNOSMĚRKOU BEZ KOLIZÍ

Vstupem hašovací funkce může být téměř libovolně dlouhá zpráva, ale výstupní kód má pevně danou délku, například u SHA-1 je to 160 bitů. Účelem hašovací funkce je, stručně řečeno, vstupní data „semlít" (slangově „*zhašovat*") složitým způsobem tak, aby výstupní vzorek tato data dostatečně dobře „jednoznačně" identifikoval – tomu říkáme **bezkoliznost**, ale aby přitom neumožnil zpětně určit původní vstup – to je tzv. **jednocestnost**. Uvozovky u slova „jednoznačně" jsou podstatné a za okamžik si vysvětlíme, co znamenají.

Bezkoliznost je zásadní pro elektronické podpisy, neboť v řadě případů se elektronicky podepisuje nikoliv samotná zpráva, ale právě jen její *hašovací hodnota* (*hašovací kód, hash,* též „*haš*"). Můžeme si to dovolit, protože úkolem hašovací funkce je právě zajistit, aby bylo výpočetně nemožné nalézt dvě různé zprávy vedoucí ke stejné haši. Ptáte se, jak je to možné, když vstupních zpráv je evidentně mnohonásobně více (například u SHA-1 je jich $2^{(2\ na\ 64)-1}$) než výstupních kódů (u SHA-1 „pouze" 2^{160})? Zákonitě přece musí vznikat ohromné množiny zpráv, které nutně vedou na tentýž výstupní kód, a vznikají tedy dokonce velmi masivní kolize!

Máte samozřejmě pravdu; podstata hašovacích funkcí je však v tom, že tyto kolize nejsme schopni nalézt! Proč tomu tak je, lze poznat z definice hašovacích funkcí – jsou to velmi složité robustní funkce, které využívají podobných technik jako šifrovací funkce. Přitom u šifrovacích funkcí jsme si zvykli, že když neznáme klíč, nemůžeme danou zprávu odšifrovat. A přesto, z informačně-teoretického hlediska to možné je, neboť běžně je v šifrovém textu příslušné množství informace k určení klíče obsaženo. Teoreticky tedy luštit lze, prakticky to ale díky složitosti příslušné úlohy neumíme.

U hašovacích funkcí je tomu podobně. A podobně jako symetrické šifry zvyšují délku klíče a složitost k dosažení vyšší bezpečnosti, hašovací funkce dělají totéž prostřednictvím délky výstupního kódu. Čím je výstupní kód delší, tím je (u kvalitních hašovacích funkcí) těžší nalézt kolize.

SAFETY FIRST – PRODLUŽUJEME!

Až dosud jsme z těch bezpečných hašovacích funkcí měli na vybranou SHA-1 a RIPEMD-160, obě dvě poskytující výsledný 160bitový hašovací kód. Ač se na nich z bezpečnostního hlediska nic nenašlo (oproti například „propadlé" MD4 a ustupující „raněné" MD5), poskytují vzhledem k tzv. narozeninovému paradoxu (viz dále) jen „bezpečnost" odpovídající symetrické šifře s 80bitovým klíčem. Naproti tomu nové hašovací funkce SHA-256, SHA-384 a SHA-512 (s délkami haše danými příslušným číslem) nabízejí bezpečnost na úrovni 128, 192 a 256 bitů, tedy na úrovni odpovídající délkám klíčů, které podporuje AES. Uvedené pravidlo srovnání bezpečnosti je dost vágní, ale použil je i americký standardizační úřad NIST při zavádění nových hašovacích funkcí.

NAROZENINOVÝ PARADOX

Dejme tomu, že máme hašovací funkci H s n-bitovým výstupním kódem. Pokud vygenerujeme 2^n+1 různých zpráv M, jistě v množině 2^n+1 hašovacích kódů H(M) nalezneme kolizi, tj. dvě různé M a M', pro něž H(M) = H(M'). Ale díky tzv. *narozeninovému paradoxu* (blíže viz např. Chip 7/98, str. 136) postačí pouze cca $2^{n/2}$ zpráv (tedy řádově méně, což je onen paradox), aby došlo ke kolizi s pravděpodobností kolem 50 %. Pro délku kódu n = 160 bitů je to 2^{80} zpráv, což by se (velmi teoreticky – někdy ve vzdálené budoucnosti) mohlo podařit pro příslušný útok vygenerovat.

Naproti tomu pro délku kódu n = 256 bitů je to 2^{128} zpráv, které už vygenerovat schopni nejsme – a doufám, že ani ve vzdálené budoucnosti nebudeme. Takže to shrňme: bezpečné hašovací funkce neumožňují nalezení kolizí žádným účinným postupem a délka jejich kódu zaručí, aby hledání kolizí narozeninovým paradoxem ($2^{n/2}$ zpráv) bylo výpočetně nezvládnutelné. Navíc za bezpečnost nových hašovacích funkcí dává ruku do ohně americký úřad NIST. Co si přát víc? Pracovní dokument definující nové funkce je veřejně k dispozici a prakticky jej můžete brát jako standard, neboť by měl být přijat co nejdříve, aby mohl být používán souběžně s AES.

NOVÉ FUNKCE VYCHÁZEJÍ Z SHA-1

Zmíněný dokument bude mít číslo *FIPS-PUB 180-2* a kromě nových funkcí SHA-256, SHA-384 a SHA-512 zahrnuje i definici stávající platné SHA-1 – jejich definice jsou si totiž velmi podobné. SHA-256 (stejně jako SHA-1) pracuje se zprávami v délce L bitů, kde $0 \le L \le 2^{64}$-1 bitů, SHA-512 zvládne délku $0 \le L \le 2^{128}$-1 bitů (včetně prázdné zprávy). Kromě délky zpracovávaných zpráv se obě tyto funkce liší ještě v délce bloku, délce slova a délce výstupního kódu, jinak mají vzorce velmi podobné. O SHA-384 →

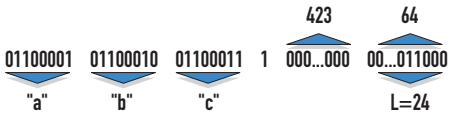
→ budeme mluvit až nakonec, protože vzniká lehkou modifikací SHA-512.

DOPLŇOVÁNÍ ZPRÁVY A DĚLENÍ NA BLOKY

Každá zpráva M je před hašováním nejprve doplněna tak, aby tvořila celistvý počet bloků, jejichž délka je 512 bitů u SHA-256 a 1024 u SHA-512. Poznamenejme, že každý blok je rozdělen na 16 slov, přičemž SHA-256 pracuje se slovy 32bitovými a SHA-512 se slovy 64bitovými. Doplnění L-bitové zprávy M provedeme jedním jedničkovým bitem a K nulovými bity (viz obr. 1).

Přitom celé číslo K ≥ 0 volíme co nejmenší tak, aby L+1+K dávalo zbytek 448 po dělení 512 (v případě

Doplnění zprávy M (řetězec "abc") na 512bitový blok pro SHA-256



Pozn.: U SHA-512 se M doplní za jedničkovým bitem 871 nulovými bity (místo 423) a L je vyjádřeno 128bitovým blokem (místo 64bitovým).

Obr. 1. Doplňování délky u SHA-256

SHA-256), resp. 896 po dělení 1024 (u SHA-512). Do zbývajících 64 bitů (u SHA-256), resp. 128 bitů (u SHA-512) vložíme právě jeden blok, který vyjadřuje číslo L v notaci BIG ENDIAN. Nyní má zpráva celistvý počet bloků (N), které označíme M^1 až M^N, kde M^1 je prvních 512 (1024) bitů zprávy.

DEFINICE POJMŮ

Než se dostaneme k popisu činnosti hašovacích funkcí, musíme zavést ještě další potřebné pojmy. Se slovy (32b/64b) se provádějí běžné *logické operace* **AND**, **OR**, **NOT**, **XOR** (s jednotlivými bity slov). Dále označme *posuv*, resp. *cyklický posuv* slova x doprava o n bitů jako **SHR(n,x)**, resp. **ROTR(n,x)**. *Aritmetické sčítání* v šíři celého slova označujeme znaménkem + (přetečení přes délku slova se zanedbává). Dále na slovech definujeme tzv. *majoritní funkci* **Maj** (*Majority*) a *funkce výběru* **Ch** (*Choice*) obvyklým způsobem,

Speciální funkce

Pro SHA-256:

Sigma_0_256(x) = (ROTR(2,x) XOR ROTR(13,x) XOR ROTR(22,x))
Sigma_1_256(x) = (ROTR(6,x) XOR ROTR(11,x) XOR ROTR(25,x))
sigma_0_256(x) = (ROTR(7,x) XOR ROTR(18,x) XOR SHR(3,x))
sigma_1_256(x) = (ROTR(17,x) XOR ROTR(19,x) XOR SHR(10,x))

Pro SHA-384 a SHA-512:

Sigma_0_512(x) = (ROTR(28,x) XOR ROTR(34,x) XOR ROTR(39,x))
Sigma_1_512(x) = (ROTR(14,x) XOR ROTR(18,x) XOR ROTR(41,x))
sigma_0_512(x) = (ROTR(1,x) XOR ROTR(8,x) XOR SHR(7,x))
sigma_1_512(x) = (ROTR(19,x) XOR ROTR(61,x) XOR SHR(6,x))

Obr. 2. Speciální funkce

tj. Maj(x,y,z) = (x AND y) XOR (x AND z) XOR (y AND z) a Ch(x,y,z) = (x AND y) XOR (NOT(x) AND z).

Kromě toho se definují čtyři *speciální funkce* pro SHA-256 a čtyři pro SHA-512 (viz obr. 2) a některé *konstanty* (najdete je na přiloženém Chip CD 8/01). Dále používáme *proměnné* ***a*** až ***h***, ***H0***^{i} až ***H7***^{i}, ***W(t)***, ***T1***, ***T2***, ..., vše slova o dané délce (32b/64b). S touto výbavou se už můžeme pustit do popisu zpracování zprávy u jednotlivých nových funkcí.

SHA-256

Nejprve inicializujeme proměnné $H0^0$ až $H7^0$ po řadě předem definovanými inicializačními hodnotami (pokud byste si chtěli algoritmus sami vyzkoušet, všechny potřebné konstanty najdete jako textový soubor v rubrice Chip Plus na Chip CD 8/01). Zprávu máme již připravenu v blocích M^i pro i = 1 až N, takže nyní budeme zpracovávat blok M^i a přitom z $H0^{i-1}$ až $H7^{i-1}$ vypočteme nové hodnoty $H0^i$ až $H7^i$. Po zpracování posledního bloku M^N bude výsledná hašovací hodnota rovna $H0^N \| H1^N \| \ldots \| H7^N$.

Zpracování bloku každého z bloků M^i pro i = 1 až N probíhá v pěti krocích:

(1) M^i rozdělíme na 16 slov: $M^i = W(0) \| \ldots \| W(15)$.

(2) Tato slova expandujeme dále pro t = 16 až 63 podle vztahu
W(t) = sigma_1_256(W(t-2)) + W(t-7)
+ sigma_0_256(W(t-15)) + W(t-16).

(3) Do proměnných a až h zkopírujeme poslední hodnoty slov H0 až H7:
$a = H0^{i-1}$, $b = H1^{i-1}$, $c = H2^{i-1}$, $d = H3^{i-1}$, $e = H4^{i-1}$, $f = H5^{i-1}$, $g = H6^{i-1}$, $h = H7^{i-1}$

(4) V následujícím cyklu ke slovům a až h přimícháváme slova W(t) a konstanty K256(t) (viz opět Chip CD 8/01) podle symbolického zápisu:
for t = 0 to 63
{ T1 = h + Sigma_1_256(e) + Ch(e,f,g) +
K256(t) + W(t);
T2 = Sigma_0_256(a) + Maj(a,b,c);
h = g; g = f; f = e; e = d + T1; d = c; c = b;
b = a; a = T1 + T2;}

(5) Vypočteme další mezivýsledky podle vztahů:
$H0^i = H0^{i-1} + a$; $H1^i = H1^{i-1} + b$; $H2^i = H2^{i-1} + c$;
$H3^i = H3^{i-1} + d$; $H4^i = H4^{i-1} + e$; $H5^i = H5^{i-1} + f$;
$H6^i = H6^{i-1} + g$; $H7^i = H7^{i-1} + h$;
a tím je popis celý.

SHA-512

Zpracování zprávy M funkcí SHA-512 se odlišuje od SHA-256 jen nepatrně – v právě popsaném postupu to znamená pouze tyto změny: inicializace se provede jinými hodnotami než u SHA-256, namísto 32bitových slov se pracuje s 64bitovými, cykly v v bodech (2) a (4) probíhají do 79 namísto 63, konstanty

Kontrolní příklady

SHA-256("abc") =
"ba7816bf8f01cfea414140de5dae2223b00361a396177a9cb410ff61f20015ad"

SHA-384("abc") =
"cb00753f45a35e8bb5a03d699ac65007272c32ab0eded1631a8b605a43ff5bed8086072ba1e7cc2358baeca134c825a7"

SHA-512("abc") =
"ddaf35a193617abacc417349ae20413112e6fa4e89a97ea20a9eeee64b55d39a2192992a274fc1a836ba3c23a3feebbd454d4423643ce80e2a9ac94fa54ca49f"

Obr. 3. Kontrolní příklady

K256(t) jsou nahrazeny konstantami K512(t) a funkce Sigma_0/1_256 a sigma_0/1_256 jsou nahrazeny funkcemi Sigma_0/1_512 a sigma_0/1_512.

SHA-384

SHA-384 se od SHA-512 liší jen jinými inicializačními hodnotami a tím, že výsledná hodnota hašovacího kódu není $H0^N \| H1^N \| \ldots \| H7^N$ (jako u SHA-512), ale jen $H0^N \| H1^N \| \ldots \| H5^N$, tj. vzniká prostým zkrácením původního výstupu na 384 bitů.

Pokud jste si popsané algoritmy zkusili naprogramovat, na obrázku 3 najdete základní kontrolní příklad pro všechny uvedené funkce.

ZÁVĚR

Je velmi pravděpodobné, že současný pracovní návrh normy FIPS PUB 180-2 neprojde žádnou zásadní změnou, zejména nikoli v definici nových hašovacích funkcí. Je také zájem, aby tato norma byla vydána v těsném závěsu za šifrovacím standardem AES, k čemuž dojde během letošního léta. Tím budeme mít k dispozici silné nástroje pro hašování i šifrování na přibližně si odpovídající bezpečnostní úrovni. Je nepochybné, že nové hašovací standardy budou implementovány souběžně s novými implementacemi schémat digitálního podpisu a standardem AES (pro generování jeho klíčů), takže se v historicky velmi krátké době pravděpodobně stanou nejrozšířenějšími hašovacími funkcemi na světě.

Vlastimil Klíma | vlastimil.klima@i.cz

INFOTIPY

Nové hašovací funkce:
► *http://csrc.nist.gov/encryption/shs/dfips-180-2.pdf*

Hašovací funkce MD, SHA, RIPEMD a technika HMAC:
► „Výživná haše", Chip 3/99;
► „Jak se melou data", Chip 4/99
(články naleznete také v elektronické podobě na *http://www.decros.cz/bezpecnost/_kryptografie.html*)