Hashovací funkce v roce 2004

(rozšířená a opravená verze), říjen 2004

Jaroslav Pinkava

PVT a.s.

1.Úvod

Z hlediska algoritmů definovaných v normách prochází v posledních létech kryptografie jako celek obdobím určitého zpřesňování a stabilizace. Výrazným příkladem toho je především oblast symetrické kryptografie - vzpomeňme přechod od DES k AES. Jedná se jak o zhodnocení objektivního navýšení výpočetního potenciálu použitelného pro útoky hrubou silou, tak i o důsledek rozvoje samotných kryptoanalytických postupů. Jako logický dopad se objevily (kromě přístupů k samotným konstrukcím algoritmů) zvýšené nároky na délku klíče (DES - 56 bitů, AES minimálně 128 bitů). Platí to ale i o bezpečnosti dalších kryptografických postupů - asymetrická kryptografie, kryptografické protokoly (např. RSA-OAEP, prokazatelně bezpečná schémata atd.).

Relativně málo se nový vývoj týkal takové oblasti kryptografie, kterou jsou hashovací funkce. Zde v posledních letech - kromě třídy *SHA-2*, kterou navrhla NSA - se objevil vlastně jen algoritmus Whirlpool v rámci aktivity Cryptonessie, a to i přesto, že původní představy organizátorů této aktivity předpokládaly podstatně širší zastoupení nových hashovacích algoritmů. Odezvu proto vyvolaly nedávné výsledky (prezentované zejména na konferenci CRYPTO 2004) týkající se hashovacích funkcí, které vzbudily pozornost i mimo kryptologickou obec.

Z těchto výsledků vyplývá, že některé hashovací funkce nesplňují, co se od nich očekávalo - týká se to algoritmů *SHA-0, MD4, MD5, HAVAL-128*, a *RIPEMD*. Ze zmíněných hashovacích funkcí algoritmy *SHA-0* a *MD4* fakticky již používány nejsou (místo *SHA-0* je používána její opravená varianta *SHA-1*), méně populární jsou funkce *HAVAL-128 a RIPEMD*.

Některé výsledky, které se týkaly kolizí známých hashovacích funkcí (Dobbertin, [18], [23]) byly již známy dříve. Na jejich základě bylo ustoupeno od používání hashovací funkce *MD4,* co se však týče dalších běžně používaných hashovacích funkcí jako *MD5*, neměl žádný z nich však bezprostřední praktické dopady resp. nebylo jejich využití zřejmé.

Hlubší zkoumání dnes používaných hashovacích funkcí má svou logiku i v návaznosti na vývoj pohledu na celkovou bezpečnost kryptografických postupů. Jestliže například (i z důvodů krátkého klíče, který nemusí déle odolávat útokům pomocí hrubé síly) přecházíme v symetrické kryptografii k schématům typu *AES* (délka klíče 128 resp. 192 resp. 256 bitů), je logické, že v tomto směru je nutné zohlednit i pohled na hashovací funkce ([6], [23]).

2. Integrita dat

Pojem integrity dat, tak jak je v kryptologii používán, úzce souvisí s pojmem kódy pro detekci chyb. Cílem příslušných postupů je zamezit modifikacím přenášených informací, ať už tyto modifikace vznikají díky chybám na přenosových cestách či díky úmyslným zásahům. Příkladem takovýchto postupů jsou kontrolní součty (checksum) detekující chyby při ukládání a přenášení datových souborů či CRC. Použití kryptografických transformací (s utajovaným klíčem) navíc zabezpečuje přenášená dat i proti útokům sofistikovanějších protivníků.

3. Vlastnosti hashovacích funkcí

Obecně se hashovací funkcí chápe zobrazení *h*, které přiřazuje zprávě jako vstupu výstup označovaný slovem hash (hodnota hashe), resp. je to zobrazení, které řetězci libovolné délky přiřazuje řetězec pevné délky. Samozřejmě v takovém případě je existence kolizí (dvojic vstupů s týmž výstupem) nevyhnutelná. Kryptografickou hashovací funkcí se pak rozumí funkce, která má navíc i určitou bezpečnostní vlastnost právě ve vztahu k možnostem vyhledávání kolizí. Předpokládá se, že popis hashovací funkce je veřejně znám, tj. samotný algoritmus není utajován.

V literatuře k hashovacím funkcím se objevuje několik typů definic, které tyto pojmy upřesňují. Např. Mao [4] uvádí následující:

V1. Praktická efektivnost: Pro dané x je výpočet $h(x)$ efektivně proveditelný (přesněji - je proveditelný v čase, který je omezen polynomiální funkcí délky vstupu x).
V2. Mixující zobrazení: Pro každý vstup x má výstupní hodnota "náhodný" charakter (autorova definice je přesnější, vyžaduje však zavedení některých dalších pojmů).
V3. Rezistance vůči kolizím: Je z výpočetního hlediska neuskutečnitelné nalézt dva vstupy x, y $(x \neq y)$, aby $h(x) = h(y)$.
V4. Rezistance vzorů: Pro danou hodnotu hashe h je výpočetně neuskutečnitelné nalézt vstupní řetězec x tak, že $h = h(x)$.

Menezes ([3]) uvádí ještě následující vlastnosti:

V5. Rezistance druhého vzoru: Je výpočetně neuskutečnitelné pro daný vstup x nalézt druhý vstupní řetězec y tak, že $h(y) = h(x)$.
Tato vlastnost se od vlastnosti V3 liší tím, že zde je jeden vstup již fixován.

V6. Rezistance vůči blízkým kolizím: Je z výpočetního hlediska neuskutečnitelné nalézt dva vstupy x, y $(x \neq y)$, tak, že $h(x)$ a $h(y)$ se liší jen v malém počtu bitů.
Blízké kolize jsou nejjednodušším příkladem zakázaných vztahů mezi výstupy hashovací funkce.

Pieprzyk ([11]) na základě vlastností V1-V5 uvádí definice dvou typů hashovacích funkcí:

OWHF - one-way hash function - *jednosměrné hashovací funkce*. Pro ně jsou splněny vlastnosti V1+V2+V4+V5;

CRHF - collision resistant hash function - *hashovací funkce rezistantní vůči kolizím*. Splňuje podmínky V1+V2+V3.

Hashovací funkce rezistantní vůči kolizím je vždy jednosměrnou hashovací funkcí. První tvrzení platí jednoduše, neboť z vlastnosti V3 plyne vlastnost V5. Jednosměrnost hashovací funkce rezistantní vůči kolizím se dokazuje sporem (podrobnosti např. v [11]). Studium přesných vztahů jednotlivých pojmů (není to triviální záležitost) je provedeno v článku [17].

Poznámka: Můžeme se také setkat s označením hashovacích funkcí jako *MDC* (Manipulation Detection Code). Existují ještě hashovací algortimy, které se navíc opírají o hodnotu (tajného) kryptografického klíče - tzv. *MAC* (Message Authentication Code). Jejich diskuse by si však zasloužila samostatný článek.

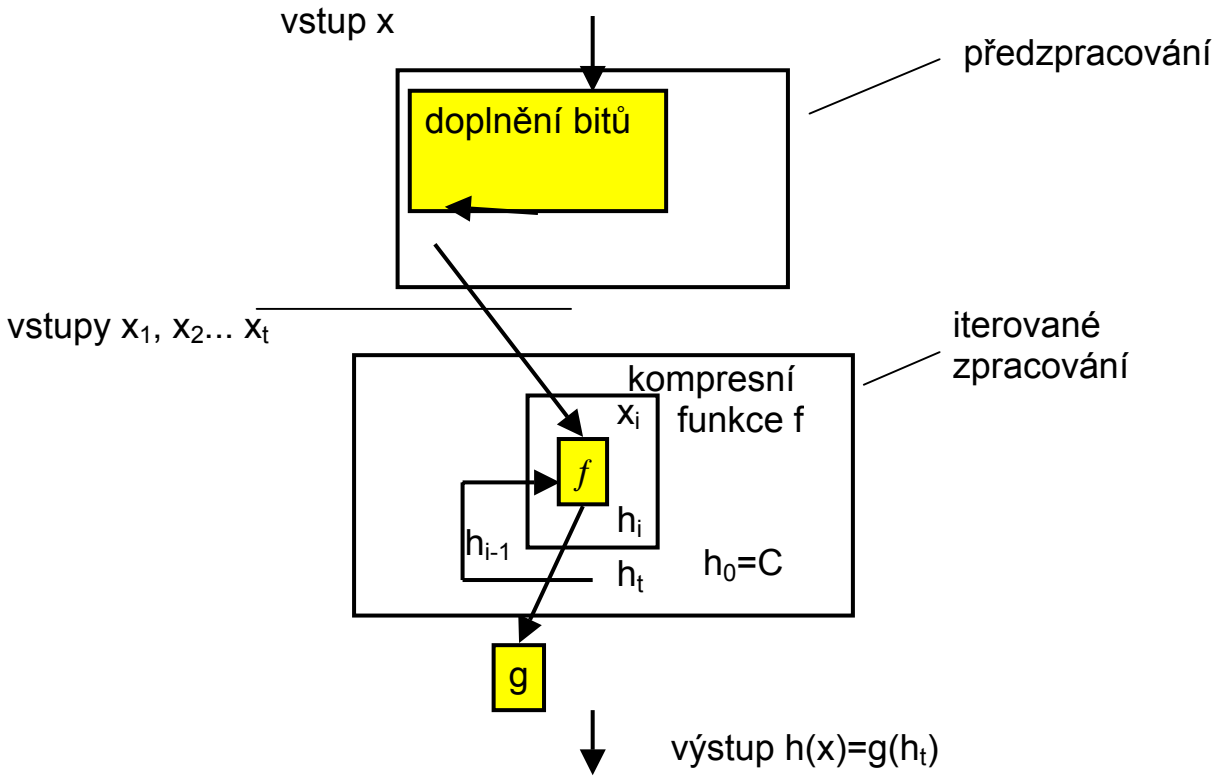
Stěžejním použitím hashovacích funkcí je oblast autentizace informací. Například ochranu autentičnosti dlouhé zprávy můžeme nahradit ochranou autentičnosti krátkého výsledku hashovací funkce. Velký datový soubor je zasílán nechráněným komunikačním kanálem a pouze spočtený výsledný hash je zaslán nějakou chráněnou cestou. Odsud se dostáváme i k použití hashovacích funkcí při tvorbě digitálních podpisů zpráv. Podepisován je pouze výsledný hash zprávy (lze samozřejmě podepisovat celou zprávu, je však otázkou efektivnost takovéhoto postupu). Tomuto výslednému hashi se zde obvykle říká otisk zprávy (message digest, fingerprint).

Jiným příkladem použití hashovacích funkcí je porovnání dvou různých textů bez rozkrytí jejich obsahu - jsou porovnávány pouze hashe těchto textů - používá se to například při verifikaci hesla. V kryptografii však existuje ještě celá řada dalších využití hashovacích funkcí (procesy pro odvození náhodně generovaných klíčů, kryptografické protokoly, existence jednosměrných hashovacích funkcí je obvykle základním argumentem při vytváření tzv. modelu náhodného oráklu - random oracle model). Pro aplikace v rámci digitálního podpisu (podepisuje se hash zpráv) je nezbytné použití CRHF, jinde (ochrana hesla) je postačující vlastnost rezistance vzorů.

Většina hashovacích funkcí vychází z využití tzv. kompresní funkce, která má pevnou délku vstupu a zpracovává shodně každý blok zprávy. Někdy se takovýmto hashovacím funkcím říká také iterované hashovací funkce. Vstup do hashovací funkce je doplněn tak, aby délka vstupu byla násobkem délky bloku. Potom je vstup x rozdělen na jednotlivé bloky x_i a hash je počítán iterativně (C je iniciační konstanta):

$$h_0 = C,$$
$$h_i = f(x_i , h_{i-1}), \quad i = 1,\ldots,t$$
$$h(x) = g(h_t).$$

Zde f je kompresní funkce, g je tzv. výstupní zobrazení (většinou je to identické zobrazení, tj. funkce jejímž výstupem je její vstup).



Doplnění bitů musí být prováděno tak, aby nemohly existovat dvě různé zprávy, které po provedeném doplnění jsou identické. Doplněk obsahuje v sobě také informaci o délce zprávy.

4. Hashovací funkce třídy MD

V článcích autorů Merkle [20] a Damgard [21] bylo ukázáno, že kompresní funkce rezistantní vůči kolizím vede k tomu, že iterovaná hashovací funkce (předešlý odstavec) je rovněž rezistantní vůči kolizím. Popsané konstrukci v předešlém odstavci se také říká konstrukce Merkle-Damgarda (odsud plyne používání zkratky MD). Na tomto základě začaly vznikat první konstrukce kryptografických hashovacích funkcí třídy MD.

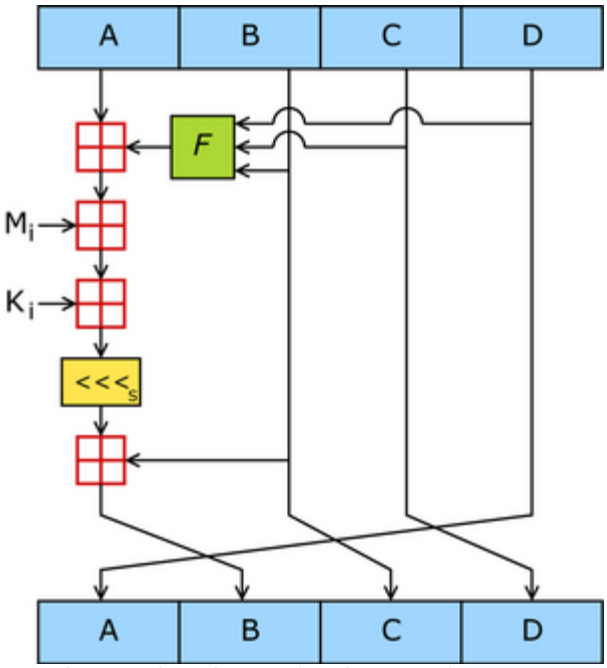
Existují tři používané (alespoň v minulosti) zástupci této třídy, autorem všech je Ronald Rivest (písmeno R v RSA). Byly zkonstruovány v létech 1989 (*MD2*), 1990 (*MD4*) a 1991 (*MD5*). Přitom *MD2* je orientována na osmibitové procesory a nezapadá tedy do rámce "dnešních" hashovacích funkcí. Délka výstupu *MD2* je 128 bitů. Algoritmus *MD4* je již podstatně rychlejší (délka výstupu je rovněž 128 bitů). Kompresní funkce zpracovává 512 bitový blok zprávy a 128 bitový blok mezivýstupu na 128 bitový blok mezivýstupu. *MD4* je považována vzhledem k existujícím kryptoanalytickým výsledkům (Dobbertin 1995 aj.) za nedostatečně bezpečnou.

Podrobný popis a zdrojový kód ke všem třem algoritmům lze nalézt v [2] . Obraťme se k hashovací funkci *MD5*. Vlastní algoritmus funguje následovně:

Výstupem *MD5* je hash v délce 128 bitů. Vstup zprávy je doplněn takovým způsobem, aby celková délka vstupu byla dělitelná 512 (v bitech). Přesněji - doplnění (padding) probíhá následovně. Zpráva je doplněna na konci nejprve jedním bitem rovným jedné. Pak je doplňována nulami tak, aby vznikl soubor o délce, která je o 64 bitů kratší než násobek 512. Zbylých 64 bitů je vyplněno číslem, které charakterizuje délku původní zprávy. Výsledek má tedy délku, která je násobkem 512.

Následující popis je převzat z [5] :

Algoritmus pracuje s blokem v délce 128 bitů (=stavový blok), který je rozdělen na 4 slova *A*, *B, C* a *D* v délce 32 bitů. V počátku algoritmu jsou hodnoty těchto slov rovné definované pevné iniciální hodnotě. Algoritmus zpracovává vždy 512 bitový blok vstupu, výsledkem je nový stavový blok (viz obrázek)



$$F(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z)$$
$$G(X,Y,Z) = (X \wedge Z) \vee (Y \wedge \neg Z)$$
$$H(X,Y,Z) = X \oplus Y \oplus Z$$
$$I(X,Y,Z) = Y \oplus (X \vee \neg Z)$$

$\oplus, \wedge, \vee, \neg$ značí operace XOR, AND, OR a NOT.

Zpracování 512 bitového vstupu probíhá ve čtyřech cyklech, každý cyklus se skládá ze 16 operací založených na nelineární funkci *F*, modulárním součtu a levé rotaci. Obrázek ilustruje průběh jedné takovéto operace. Jsou použity čtyři možné funkce *F*, v každém cyklu je použita jiná.

Pro algoritmus *MD5* byly nalezeny kolize již dříve (Dobbertin [18], den Boer, Bosselaers [28]), ale byly to kolize ve specifické podobě, které nemají přímý dopad na aplikace.

5. Další známé hashovací funkce

V roce 1992 navrhli Zheng, Pieprzyk a Seberry hashovací funkci HAVAL. Sestává z několika variant (délka výstupu mezi 128 a 256 bity, pracuje v 3 či 4 či 5 cyklech).

NIST opublikoval několik variant hashovacích funkcí jako normu FIPS, které vycházely z *MD4*. První *SHA* (Secure Hash Algorithm) algoritmus byl opublikován v roce 1993 (FIPS 180, květen 1993). Velikost výstupu této hashovací funkce je 160 bitů. V roce 1995 objevil NIST slabinu v *SHA-0*, to vedlo k vydání nové verze pod názvem *SHA-1* (FIPS 180-1, duben 1995) - byla zde přidána jedna nová rotační operace.

Dále NIST opublikoval v roce 2002 tři nové hashovací algoritmy pod názvy *SHA-256*, *SHA-384* a *SHA-512* ([23]). V prosinci 2003 byl přidán ještě hashovací algoritmus *SHA-224*. *SHA-224* a *SHA-256* pracují s osmi 32-bitovými mezivýstupy a jejich kompresní funkce zpracovává blok zprávy v délce 512 bitů a mezivýstup v délce 256 bitů. *SHA-384* a *SHA-512* pracují s osmi 64 bitovými mezivýstupy a jejich kompresní funkce zpracovává bloky zpráv v délce 1024 bitů a mezivýstup v délce 512 bitů.

Jiná opravená verze *MD4* s názvem *RIPEMD* byla vyvinuta v Evropě v rámci projektu RIPE (EEC-RACE, 1995). Vzhledem k výsledkům Dobbertina vznikly posléze dvě nové verze: *RIPEMD-128* a *RIPEMD-160* (1996).

V rámci projektu Cryptonessie (viz informace na stránkách Crypto-Worldu) byl navržen algoritmus *Whirlpool*. Algoritmus vychází z obdobných konstrukčních principů jako *AES*, jeho výstupem je 512 bitový hash.

ISO norma (ISO/IEC 10118-3) obsahuje následující algoritmy: *RIPEMD-128, RIPEMD-160, SHA-1, SHA-256, SHA-384, SHA-512* a *Whirlpool*.

Dalšími relativně známými hashovacími algoritmy jsou např.: *Snefru*, *Subhash* a *Tiger*.

Poměrně úplný přehled existujících hashovacích funkcí včetně referencí může čtenář nalézt v [13]. Pěkný dokument (Ilustrovaný úvod do problematiky kryptografických hashovacích funkcí) je zpracován v [25].

6. Některé útoky na hashovací funkce

Z hlediska bezpečnosti hashovacích funkcí musí být vyhledávání kolizí obtížnou úlohou. Musíme předpokládat, že potenciální útočník zná hashovací algoritmus a může provádět útoky s možností libovolně volit zprávu, pro kterou je následně spočten hash (adaptive chosen message attack).

Útoky na hashovací funkce lze třídit dle následujících typů:

- útok hledající kolize - hledám dvě různé zprávy X, Y tak, že $h(X) = h(Y)$;
- útok hledající vzor - pro náhodnou hodnotu Y (v délce n) hledám zprávu X tak, že $h(X) = Y$;

- útok hledající druhý vzor - při dané zprávě X, hledám zprávu Y (různou od X) tak, že $h(X) = h(Y)$.

Jsou také studovány následující rozšíření těchto útoků ([19], [22]):

- útok hledající tzv. K-kolize pro $K \geq 2$ - hledám K různých zpráv $X(j)$ tak, že $h(X(1)) = \ldots = h(X(K))$.
- útok hledající vzory (druhé vzory) pro $K \geq 1$ - při daném Y (či Q takovém, že $h(Y) = Q$), hledám K různých zpráv $X(j)$ tak, že $h(X(1)) = \ldots = h(X(K)) = Y$ (a přitom $X(j)$ nejsou rovny Y).

Jedním z útoků, který je (v zásadě) aplikovatelný vůči všem hashovacím funkcím je tzv. narozeninový útok.

Tento útok vychází z jedné známé úlohy z teorie pravděpodobnosti - narozeninový paradox. Je formulována takto: Jaký musí být minimální počet žáků ve třídě, aby pravděpodobnost, že alespoň dva žáci se narodili ve stejný den, byla větší než jedna polovina? Odpovědí na tuto otázku je číslo 23. Tento výsledek svým způsobem protiřečí intuici (máme 365 dní v roce). Úspěšnost narozeninového útoku je dána délkou výstupu n hashovací funkce. Pro určitý počet N vstupních náhodných hodnot spočteme jejich hashe. Potom pravděpodobnost, že získáme kolizi (dva stejné hashe) je rovna jedné polovině (přibližně), pokud

$$N \sim 1{,}1774 \sqrt{(2^n)}$$

Vzhledem k tomu, že *MD5* má délku výstupu 128 bitů, je třeba řádově 2^{64} náhodných pokusů k nalezení takovéto kolize. Mimochodem - v březnu 2004 byl zahájen projekt MD5CRK [6] , jehož cílem bylo právě provedení tohoto útoku. Důvodem bylo poukázat na nedostatečnou délku výstupu hashovací funkce *MD5*. Ovšem v důsledku výsledku čínských matematiků byl v srpnu 2004 projekt ukončen. Jednoduchým dopadem tohoto útoku je požadavek dvojnásobné délky n výstupu hashovací funkce (ve smyslu CRHF) oproti nárokům na délku klíče symetrického šifrovacího algoritmu. Je to nezbytné pro dosažení (zhruba řečeno) obdobné odolnosti proti útoku hrubou silou. Na druhou stranu pro OWHF zůstáváme u požadavku na délku n výstupu hashovací funkce nelišícího se od požadavku na délku klíče symetrického šifrovacího algoritmu. Záleží tedy na tom, na co hashovací funkci používáme.

Narozeninový útok nepoužívá žádnou z vlastností hashovací funkce, která souvisí s jeho vnitřní strukturou. Existuje varianta útoku - *meet-in-the-middle attack*. Tato varianta již není univerzální, je aplikovatelná vůči hashovacím funkcím, které používají nějakým způsobem postupy, při kterých dochází k řetězení bloků, tj. kompresní funkce je snadno invertovatelná. Tento útok umožňuje útočníkovi zkonstruovat podvrženou zprávu, která má hash volený útočníkem. Útočník generuje k_1 variant první části podvržené zprávy. Vyjde z iniciační hodnoty a pokračuje až do nějakého mezilehlého stavu algoritmu. Dále vygeneruje k_2 variant druhé části podvržené zprávy. Začne s danou cílovou hodnotou hashe a vrací se zpět do mezilehlého stavu. Pravděpodobnost shody v tomto mezilehlém stavu je dána opět pravděpodobností úspěchu narozeninového útoku.

Zpráva nechť je rozdělena na dvě části $m = (m_1, m_2)$, pak její hash je počítán ve dvou krocích. Nejprve spočteme $h_1 = H(m_1, C)$, kde C je iniciační vektor (známé číslo), H je

příslušná hashovací funkce. Dále pak spočteme $h = H(m_2, h_1)$. Nyní nechť útočník chce nalézt podvrženou zprávu $m' = (m_1', m_2')$, která koliduje s hashem h. Útočník zvolí nyní k_1 možností pro m_1' a k_2 možností pro m_2'. Jsou tedy dvě množiny:

$$\{m_{1,i}' : i = 1,...,k_1\} \quad \text{a} \quad \{m_{2,i}' : i = 1,...,k_2\}.$$

Útočník spočte k_1 variant $H(m_{1,i}', C)$ a k_2 variant $H^{-1}(m_{2,i}', h)$. Pravděpodobnost shody ve spočtených množinách hodnot je pak táž jako v narozeninovém útoku - pokud se H chová skutečně jako náhodná funkce. Je dána vzorcem

$$P = 1 - exp(-k_1.k_2/2^n).$$

Vzhledem k podobnostem konstrukcí blokových šifer a hashovacích algoritmů - iterovaná schémata - je za důležitý zdroj potenciálních útoků proti hashovacím funkcím považována *diferenciální kryptoanalýza* (viz např. [12], [14] a [15] - ale existuje další dlouhá řada odkazů). Tyto útoky jsou směrovány na vnitřní vlastnosti kompresní funkce.

7. Crypto 2004 a útoky na hashovací funkce

Již v průběhu roku se objevily některé informace o nových výsledcích ve vztahu k bezpečnosti populárních hashovacích funkcí (např. Biham [24]), ale hlavní výsledky byly prezentovány na tradiční konferenci CRYPTO 2004 pořádané IACR každoročně v Santa Barbaře v Kalifornii (15.8- 19.8 2004, http://www.iacr.org/conferences/crypto2004/). Popisované útoky ([19], [10], [7]) vyúsťují v nalezení některých kolizí pro známé hashovací funkce, tj. nalezení řetězců X a Y takových, že $h(X) = h(Y)$.

Praktickou bezpečnost samozřejmě podstatně více ohrožují "smysluplné" kolize, tj. takové kolize, kde řetězce X a Y nejsou jen nějakou posloupností bitů, ale X a Y jsou řetězce čitelného textu. Následkem takovéto situace je již samozřejmě bezprostřední ohrožení deklarované bezpečnosti. Bylo by například možné vytvářet dvojici zpráv s týmž otiskem a odlišným významem. Podpis příslušného otisku zprávy by pak mohl být interpretován dvěma různými způsoby a pro následné soudní (např.) rozhodování by bylo problematické stanovit, která z těchto interpretací je správná.

Útoky prezentované na konferenci CRYPTO 2004 však naštěstí zatím tak daleko nesahají, tj. vlastnost V5. - rezistance druhého vzoru - narušena nebyla. Přístup každé ze tří skupin autorů je odlišný (výsledky byly prezentovány jak v rámci hlavního pořadu konference, tak i na neformální tzv. rump session), všechny tři ale vychází z aplikací některých technik diferenciální kryptoanalýzy. Eli Biham soustředil svou práci na nalezení kolizí pro *SHA-0*. Antoine Joux ve své práci popisuje zajímavý výsledek spočívající v tvrzení, že nalezení vícenásobných kolizí není obtížnější než nalezení jedné kolize (pro algoritmy Feistelova typu, tj. iterované algoritmy). Jouxova práce vyústila v nalezení kolizí pro *SHA-0* ([26]). Čtyři čínští kryptografové prezentovali některé konkrétní kolize (především pro *MD5*, ale také *HAVAL, MD4* a *RIPEMD*). Čínské výsledky měly být též součástí vystoupení hlavního pořadu konference, ale pro neúplnost a nesrozumitelnost popisu použitých technik byly předneseny pouze v rámci rump session.

Kritické výsledky se tedy týkají především algoritmu *MD5*. Mimochodem - laboratoře RSA doporučily již v roce 1996 přechod na algoritmus *SHA-1*.

7.1. Eli Biham a SHA-0 (resp. SHA-1)

Eli Biham (ve spolupráci s Rafi Chenem) prezentoval své výsledky již na konferenci SAC počátkem srpna 2004 ([24]). V roce 1998 popsali Chabaud a Joux ([27]) útok, který nalézá kolize pro *SHA-0* a má výpočetní složitost 2^{61} (konkrétní kolize našli pro *SHA-0* redukované na 35 cyklů). Útok se opírá o použití diferenciální kryptoanalýzy. Zprávy jsou konstruovány tak, aby zde existovaly specifické diference a efekt těchto diferencí je likvidován během několika cyklů. Eli Biham ukázal na konferenci SAC jak tyto výsledky lze dále prohloubit. Biham nalezl kolize pro *SHA-0* redukované na 65 cyklů a blízké kolize pro plnou kompresní funkci *SHA-0*, kde se shoduje 142 ze 160 bitů.

V rámci konference CRYPTO 2004 bylo předneseno další rozvinutí těchto výsledků ([10]). Základní výsledky původní verze článku se shodují s výsledky, které byly prezentovány na konferenci SAC. Po napsání této původní verze získali autoři další nové výsledky. Vylepšení jsou následující. Pro *SHA-0* autoři konstatují, že tato hashovací funkce nesmí být používána v aplikacích, kde je existence kolizí zranitelností. Byl nalezen postup, který umožňuje využití existence blízkých kolizí pro nalezení vlastních kolizí (postup se ukázal být užitečný pro situace, kde původně kolize nešlo najít, např. pro *SHA-0* redukované na 50 cyklů). Také analýza *SHA-1* (redukované verze) byla prohloubena. Bylo nalezeno milion kolizí pro *SHA*-1 redukované na 34 cyklů, přitom některé z kolizí jsou částečně "smysluplné" (obsahují pouze písmena v ASCII kódu či dokonce části textu v angličtině). Blízké kolize byly nalezeny pro *SHA-1* redukované na 45 cyklů. Autoři však konstatují, že z hlediska současného stavu výzkumu neočekávají, že se jim podaří rozbít algoritmus *SHA-1*.

7.2. Anthony Joux a vícenásobné kolize

A. Joux nalezl na základě svých teoretických výsledků konkrétní kolizi pro SHA-0 ([26]). Výpočetní složitost útoku je zhruba 2^{51}.

Především však Jouxovy výsledky vrhají stín pochybnosti na správnost dosavadní konstrukce hashovacích funkcí (MD konstrukce - iterace kompresní funkce). Dle jeho výsledků nelze tímto postupem dost dobře aproximovat chování teoretického modelu náhodné funkce. Jak autor ukázal, tak nalezení vícenásobných kolizí je v iterovaném modelu výpočetně význačně méně náročné (než by odpovídalo teoretickému modelu náhodné funkce).

Poznámka: V době psaní tohoto článku jsem neměl přístup k originálnímu materiálu [19] a bylo možné vycházet jen ze zprostředkovaných popisů (např. v [22] a jinde) - JP.

7.3. Zveřejněné informace o výsledku čínských matematiků

Materiál článku [7] autorů je stručný a - jak již bylo řečeno výše - nepříliš kvalitně zpracovaný. V zásadě jen obsahuje získané kolize pro algoritmy *MD5*, *HAVAL-128*, *MD4* a *RIPEMD* spolu s komentářem, že obdobné kolize lze získat s dostatečně vysokou pravděpodobností i pro algoritmy *SHA-0* a *HAVAL-160*. Samotná metoda, kterou čínští matematici použili, zde popsána není.

Poznámka: V původní verzi článku se autor chtěl pro zjednodušení popisu obejít bez užití symbolů pro iniciační vektor (pro jednodušší odkaz na základní vlastnosti hashovacích funkcí) a i díky tomu vznikly nepřesnosti ve formulacích.

Následující popis sleduje již plně výklad autorů (pro *MD5* a s malým komentářem). Pokud se týká algoritmu *MD5*, autoři nalezli mnoho kolizí (pro originální iniciační hodnotu a podle autorů lze to provést i pro libovolnou iniciační hodnotu) odvozených ze dvou zpráv v délce 1024 bitů v podobě :

$$h(m_1, n_{1i}) = h(m_2, n_{2i}) \qquad (*)$$

kde h je hashovací funkce *MD5*. Přitom

$$m_2 = m_1 + A1$$
$$n_{2i} = n_{1i} + A2,$$

kde $A1 = (0,0,0,0,2^{31},\ldots,2^{15},\ldots,2^{31},0)$ a $A2 = (0,0,0,0,2^{31},\ldots,-2^{15},\ldots,2^{31},0)$, v obou případech nenulové pozice jsou na místech 4,11 a 14. Po nalezení nejprve vhodných m_1 a m_2 byly spočteny hodnoty (výpočetně již s výrazně menšími nároky) n_{1i} a n_{2i} a to tak, aby platila rovnost (*).

Autoři říkají, že postup lze použít pro libovolnou danou iniciační hodnotu. V článku dále následují příklady dvou dvojic 1024 bitových zpráv, které vedou ke kolizím (téže hodnotě hashe). Zprávy se přitom v některých bitech shodují. Uvádíme první dvojici zpráv v podobě, v jaké byla uvedena v článku:

m_1	2dd31d1	c4eee6c5	69a3d69	5cf9af98	87b5ca2f	ab7e4612	3e580440	897ffbb8
	634ad55	2b3f409	8388e483	5a417125	e8255108	9fc9cdf7	f2bd1dd9	5b3c3780
n_1	d11d0b96	9c7b41dc	f497d8e4	d555655a	c79a7335	cfdebf0	66f12930	8fb109d1
	797f2775	eb5cd530	baade822	5c15cc79	ddcb74ed	6dd3c55f	d80a9bb1	e3a7cc35
m_2	2dd31d1	c4eee6c5	69a3d69	5cf9af98	7b5ca2f	ab7e4612	3e580440	897ffbb8
	634ad55	2b3f409	8388e483	5a41f125	e8255108	9fc9cdf7	72bd1dd9	5b3c3780
n_2	d11d0b96	9c7b41dc	f497d8e4	d555655a	479a7335	cfdebf0	66f12930	8fb109d1
	797f2775	eb5cd530	baade822	5c154c79	ddcb74ed	6dd3c55f	580a9bb1	e3a7cc35

h	9603161f	f41fc7ef	9f65ffbc	a30f9dbf

Vyznačeny podtržením jsou rozdíly obou zpráv.

Pokud se obrátíme k definicím z paragrafu 3, vidíme, že je narušena vlastnost V3, tj. rezistance vůči kolizím. Za zmínku stojí skutečnost, že zprávy, které jsou zdrojem kolize, se liší jen ve velmi malém počtu bitů (vlastnost V3 lze tedy v tomto smyslu i poněkud upřesnit).

Na základě výsledku čínských matematiků je tedy možné zkonstruovat dvě zprávy, které se liší jen ve velmi malém počtu bitů a mají přitom týž hash (a tedy pokud budou elektronicky podepsány, budou mít i týž elektronický podpis).

Analýza zneužití získaných výsledků potenciálním útočníkem je ztížena neúplností popisu získaných výsledků. Dle některých diskusí na Internetu existují možnosti prohloubení popsaných útoků. Teprve po celkovém popisu a vyhodnocení použitých postupů bude možné zhodnotit existující rizika.

8. Hashovací funkce - co dál

Lze očekávat další rozpracování dosud získaných výsledků. V oblasti kryptografie, která se týká hashovacích funkcí dochází v současnosti tedy k posunům z hlediska nároků na bezpečnost. Např. algoritmus MD5 bude nepochybně již jen vyklízet své pozice.

Publikovaný v tomto roce potenciální útok proti *SHA-1* (Eli Biham, [10]) je efektivní pouze pro redukovanou variantu *SHA*-1 (36 cyklů namísto plných 80, samozřejmě redukovaná varianta se nepoužívá). Samotná NIST ve svém prohlášení dává *SHA-1* dobu života do roku 2010 a doporučuje používat algoritmy s větší délkou hashe (třída hashovacích algoritmů *SHA-2*, tj. *SHA-224, SHA-256, SHA-384* a *SHA-512*). Z odborných kruhů však také zaznívá volání po hlubší analýze hashovacích funkcí této třídy.

Zajímavý je návrh Schneiera na uspořádání veřejného konkurzu na nové hashovací funkce (obdoba konkurzu na *AES*). Metoda, pomocí které byly hashovací funkce třídy *SHA* vytvářeny, zůstává totiž pod pokličkou NSA. Schneier se domnívá, že veřejná kontrola napomůže větší bezpečnosti.

Literatura:

[1] RSA FAQ, http://www.rsasecurity.com/rsalabs/node.asp?id=2152
[2] RFC 1319, RFC 1320, RFC 1321, např. : http://www.cert.dfn.de/eng/resource/rfc/rfc-tit.html#TITL
[3] Menezes, Alfred; van Oorschot, P. Vanstone, S.;: Handbook of Applied Cryptology, http://www.cacr.math.waterloo.ca/hac
[4] Mao, Wenbo: Modern Cryptology, Theory and Practice, Prentice Hall 2003
[5] http://en.wikipedia.org/wiki/Md5sum
[6] http://www.md5crk.com/
[7] Xiaoyun Wang, Dengguo Feng, Xuejia Lai a Hongbo Yu *http://eprint.iacr.org/2004/199.pdf*
[8] Hash Collision Q&A, Cryptography Research, http://www.cryptography.com/cnews/hash.html
[9] Felten, Edward: An Ilustrated Guide to Cryptographic Hashes http://www.unixwiz.net/techtips/iguide-crypto-hashes.html
[10] Biham, Eli, Chen, Rafi: Near Collisions of SHA-0, Crypto 2004 http://www.cs.technion.ac.il/users/wwwb/cgi-bin/tr-get.cgi/2004/CS/CS-2004-09.ps.gz
[11] Piepryzk, Jozef; Hardjono, Thomas; Seberry, Jennifer: Fundamentals of Computer Security, Springer 2003
[12] Birjukov, Alex; Lano, Joseph; Preneel, Bart: Cryptanalysis of the Alleged SecurID Hash Function (extended version), http://eprint.iacr.org/2003/162.pdf
[13] http://planeta.terra.com.br/informatica/paulobarreto/hflounge.html
[14] Daemen, Joan: Cipher and Hash Function Design, Strategies based on linear and differential cryptanalysis http://www.esat.kuleuven.ac.be/~cosicart/ps/JD-9500/
[15] Contini, Scott; Yin, Yiqun Lisa: Improved Cryptanalysis of SecurID. http://eprint.iacr.org/2003/205.pdf
[16] Preneel, Bart; Leuven, K.U.: Hash function (Version 4, May 2004)
[17] Rogaway, P; Shrimpton, T.: Cryptographic hash function basics: Definitions,

implications, and separations for preimage resistace, second-preimage resistance and collision resistance, Fast Software Encryption, Lecture Notes in Computer Science, Springer-Verlag 2004
[18] Dobbertin, Hans: Cryptanalysis of MD4, Journal of Cryptology 1998, No.11
Cryptanalysis of MD5 Compress,
http://www.cs.ucsd.edu/users/bsy/dobbertin.ps
[19] Joux, A.: Multicollisions in iterated hash functions, applications to cascaded constructions. Crypto 04, LNCS 3152, pp.306-316
[20] Merkle, R.: One-way hash functions and DES, Crypto 89, LNCS 435, pp. 428-446
[21] Damgard, I.: A design principle for hash functions. Crypto 89, LNCS 435, pp. 416-427
[22] Lucks, Stefan: Design Principles for Iterated Hash Functions,
http://eprint.iacr.org/2004/253.pdf
[23] National Institute of Standards and Technology (NIST). Secure hash standard, FIPS 180-2, August 2002
[24] Biham, Eli: New Results on SHA-0 and SHA-1, SAC 2004,
http://www.cs.technion.ac.il/~biham/Reports/Slides/invited-talk-sac-2004.ps.gz
[25] Steve Friedl: An Illustrated Guide to Cryptographic Hashes,
http://www.unixwiz.net/techtips/iguide-crypto-hashes.html
[26] Kolize nalezená Jouxem pro SHA-0,
http://www.mail-archive.com/cryptography@metzdowd.com/msg02554.html
[27] Chabaud, Florent; Joux, Antoine: Differential Collisions in SHA-0, CRYPTO 98, LNCS 1462, pp. 56-71, Springer Verlag 1999
[28] den Boer, B.; Bosselaers, Antoon: Collision for the Compression Function of MD5, Eurocrypt 1993