Moderni napadi na zgoščevalno funkcijo MD5

projekt pri predmetu Kriptografija in teorija kodiranja II

Aljaž Košmerlj

22. julij 2008

Povzetek

Glavna tema te seminarske naloge so moderni pristopi za napade na zgoščevalno funkcijo MD5, ki so bili razviti od leta 2004 naprej. Na kratko predstavimo razvoj pred letom 2004, potem pa podrobno opišemo diferenčni napad profesorice Wang in sodelavcev, ki predstavlja osnovo vsem kasnejšim pristopom. Opišemo tudi tehniko tuneliranja, ki je pomembna teoretična nadgradnja osnovnega diferenčnega napada.

Kazalo

Poglavje 1

Uvod

1.1 Kriptografska zgoščevalna funkcija MD5

Kriptografske zgoščevalne funkcije že dolgo predstavljajo nepogrešljivo orodje v računalniški varnosti. Uporablja se jih v mnoge različne namene: za pridobivanje izvlečkov besedil, s katerimi potrdimo nihovo pristnost, za shranjevanje gesel, za testiranje neoporečnosti datotek ipd. Zaradi obsežne uporabe, predvsem z razvojem večjih omrežij, je bilo razvitih več različnih algoritmov zanje, ki so morali zadoščati strogim standardom o varnosti in hitrosti.

Ena najbolj znanih zgoščevalnih funkcij je, tako imenovana, MD5, ki jo je razvil profesor Ronald Rivest leta 1991. Kot pove ime, je MD5 peta zgoščevalna funkcija iz MD ("Message Digest") serije. Razvita je bila kot nadomestilo za, leta 1990 razvito, MD4, za katero so leta 1991 analitično odkrili šibkosti. Po objavi v RFC 1321 [1] se je zelo razširila, celo do te mere, da imajo nekateri operacijski sistemi (predvsem tisti osnovani na Unixu), že vgrajena orodja za izračun MD5. Zelo pogosto se MD5 uporablja za dokazovanje integritete programske opreme, recimo za izračun kontrolnih vsot paketov operacijskega sistema Linux.

S stališča varnosti je MD5 relativno dolgo kljubovala napadom. Čeprav so v devedesetih letih prejšnjega stoletja odkrili določene šibkosti (1993 - pseudokolizija, 1996 - kolizija kompresijske funkcije), je do pravega preboja prišlo šele leta 2004, ko je na konferenci CRYPTO 2004 skupina kitajskih raziskovalcev predstavila prva prava trčenja za MD5, ki naj bi jih odkrili z računsko izvedljivim analitičnim napadom na MD5. Algoritem za napad so predstavili naslednje leto na konferenci Eurocrypt 2005. V sledečih letih je bil ta napad deležen precej izboljšav in je sedaj možno trčenja za MD5 najti na navadnem prenosnem računalniku v nekaj minutah. Namen te seminarske naloge je predstaviti razvoj modernih napadov na MD5, ki so nastali od leta 2004 naprej.

1.2 MD5 sorodne zgoščevalne funkcije

MD5 je bila razvita kot varnejša naslednica MD4 in je zato zgrajena na istih principih. MD4 je kot osnova služila še mnogim drugim zgoščevalnim funkcijam kot so RIPEMD, HAVAL in družini SHA. Po osnovni obliki so si tako vse zelo podobne: sestavljene so iz osnovnih aritmetičnih in logičnih operacij in v enem ali večih preletih poljubno dolgega besedila izačunajo izvleček fiksne predpisane dolžine. Ta podobnost je pomembna, ker so zaradi nje principi napadov na eno teh funkcij uporabni tudi na ostalih, kar velja tudi za napade opisane v nadaljevanju.

Poglavje 2

Algoritem MD5

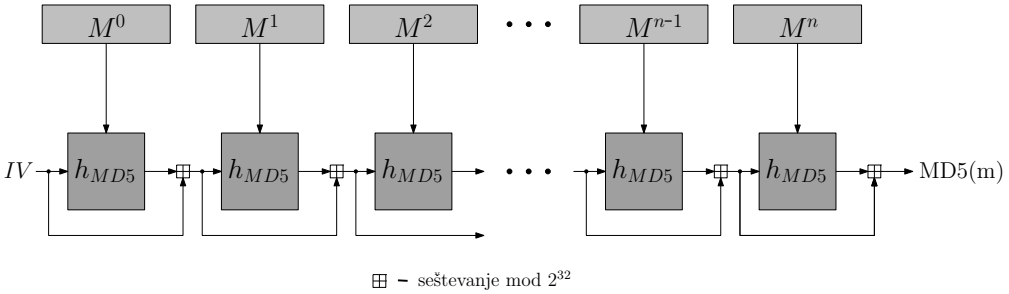
MD5 prejme sporočilo poljubne dolžine in zanj izračuna 128 biten izvleček. Sporočilo se najprej podaljša na dolžino (v bitih) kongruentno 448 po modulu 512. Temu je nato dodana 64 bitna predstavitev dolžine besedila, tako da je dolžina besedila deljiva s 512 bitov in se ga da razdeliti na 512 bitne bloke. 128 bitni izvleček posameznega bloka se izračuna samostojno s kompresijsko funkcijo $h_{MD5}(IV, B)$. Izhod funkcije na enem bloku se prišteje inicializacijskemu vektorju za ta krog, ta vsota pa se nato uporabi kot inicializacijski vektor za izračun izvlečka naslednjega bloka in tako naprej preko celotnega besedila. Inicializacijski vektor za prvi blok je sicer lahko poljuben, vendar je od njega odvisna varnost celotnega postopka, zato je ponavadi fiksno določen. Rezultat kompresijske funkcije na zadnjem bloku je izvleček celotnega besedila. Za shematski prikaz glej sliko 2.1. Zapisano simbolno:

$$MD5(m) = MD5(m') = MD5(M^0|M^1|\ldots|M^n) =$$
$$h_{MD5}(h_{MD5}(\ldots h_{MD5}(h_{MD5}(IV, M^0), M^1), \ldots), M^{n-1}), M^n)$$

Kjer je $m' = M^0|M^1|\ldots|M^n$; $|M^i = 512$, podaljšano besedilo m.

Jedro algoritma MD5 je torej kompresijska funkcija, ki prejme izvleček prejšnjega bloka (oz. inicializacijski vektor na začetku) in izračuna izvleček trenutnega bloka M. M v ta namen razdelimo na 16 32-bitnih besed $(M_0, M_1, \ldots, M_{15})$ Kompresijska funkcija je sestavljena iz 64 operacij (korakov) urejenih v 4 kroge po 16 izračunov. Stanje algoritma kompresijske funkcije določajo štirje 32-bitni registri (besede), ki jih na začetku nastavimo na posamezne besede 128-bitnega inicializacijskega vektorja:

$$\begin{aligned} R_{-4} &= IV_0 \\ R_{-3} &= IV_3 \\ R_{-2} &= IV_2 \\ R_{-1} &= IV_1 \end{aligned}$$



Slika 2.1: Shema algoritma MD5.

Na vsakem koraku se iz predhodnjih štirih registrov izračuna naslednji:

$$R_i = R_{i-1} + (f_i(R_{i-1}, R_{i-2}, R_{i-3}) + R_{i-4} + K_i + W_i)^{\lll s_i}$$

Kjer je:

- vse seštevanje opravljeno po $mod\ 2^{32}$
- W_i ustrezna beseda bloka M, izbrana po pravilu:

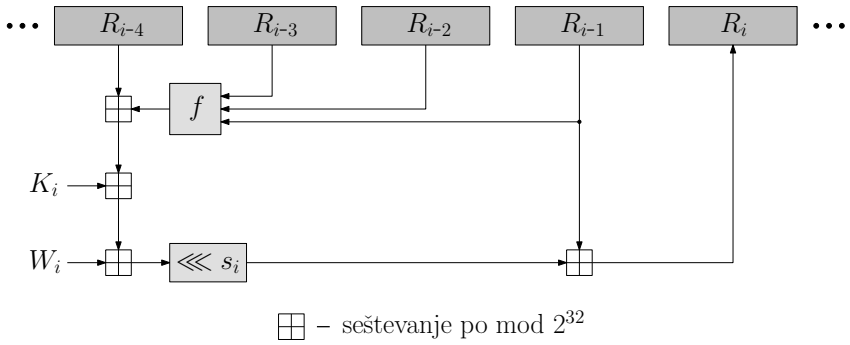
$$W_i = \begin{cases} M_i; & \text{za } 0 \leq i \leq 15 \\ M_{1+5i \bmod 16}; & \text{za } 16 \leq i \leq 31 \\ M_{5+3i \bmod 16}; & \text{za } 31 \leq i \leq 47 \\ M_{7i \bmod 16}; & \text{za } 48 \leq i \leq 63 \end{cases}$$

- K_i koračno odvisna aditivna konstanta (podrobnosti v [1])
- $\lll s_i$ krožni pomik v levo za s_i bitov; s_i je koračno odvisna konstanta (podrobnosti v [1])
- f_i ena štirih nelinearnih bitnih logičnih funkcij:

$$f_i = \begin{cases} F(x,y,z) & = (x \wedge y) \vee (\neg x \wedge z); & \text{za } 0 \leq i \leq 15 \\ G(x,y,z) & = (x \wedge z) \vee (y \wedge \neg z); & \text{za } 16 \leq i \leq 31 \\ H(x,y,z) & = x \oplus y \oplus z; & \text{za } 31 \leq i \leq 47 \\ I(x,y,z) & = y \oplus (x \wedge \neg z); & \text{za } 48 \leq i \leq 63 \end{cases}$$

Po štiriinšestdesetem koraku funkcija vrne nov inicializacijski vektor (oz. izvleček če gre za zadnji blok) $IV = IV_0|IV_1|IV_2|IV_3$:

$$\begin{aligned} IV_0 &= R_{-4} + R_{60} \\ IV_3 &= R_{-3} + R_{61} \\ IV_2 &= R_{-2} + R_{62} \\ IV_1 &= R_{-1} + R_{63} \end{aligned}$$



Slika 2.2: Shema kompresijske fukcije

kjer + spet pomeni seštevanje po modulu 2^{32}.

Očitno je, da so za izračun kompresijske funkcije v i-tem koraku potrebni le registri $R_{i-4}, R_{i-3}, R_{i-2}, R_{i-1}$ - tako lahko v spominu vedno hranimo le zadnje štiri registre (shematičen prikaz na sliki 2.2).

Poglavje 3

Napadi na MD5

3.1 Razvoj pred letom 2004

Leta 1993, dve leti po objavi MD5, sta Boer in Bosselaers odkrila pseudo-kolizijo: eno sporočilo, za katerega, za dva različna nabora začetnih vrednosti algoritma, dobimo isti izvleček. To je pokazalo, da je efekt plazu v MD5 šibkejši, kot je bilo prej znano. Podrobnosti v [2].

Tri leta kasneje je na konferenci Eurocrypt'96 Dobbertin predstavil kolizijo dveh različnih 512-bitnih sporočil za določeno fiksno vrednost IV. Opis napada je objavljen v [3].

Ker je izvleček dolg le 128-bitov, je podjetje CertainKey Cryptosystems, marca leta 2004, začelo projekt MD5CRK, v katerem so hoteli s porazdeljenim rojstnodnevnim napadom poiskati trčenje v MD5. Projekt se je končal avgusta 2004, ko so bila objavljena trčenja dobljena z analitičnim napadom (glej naslednji razdelek).

3.2 Osnovni diferenčni napad

Leta 2004 je skupina kitajskih kriptografov, s profesorico Xiaoyun Wang na čelu, objavila seznam trčenj za več znanih zgoščevalnih funkcij, med njimi MD5 (točen seznam v [4]). Algoritem, s katerim so odkrili ta trčenja, pa so predstavili naslednjega leta na konferenci Eurocrypt 2005. Napad za dan inicializacijski vektor IV_0 poišče dve 1024-bitni besedili, ki imata enak MD5 izvleček.

3.2.1 Diference

Napad, ki so ga razvili, je diferenčen napad. Diferenčna kriptografija je veja kriptografije, ki preučuje kako (majhne) razlike v čistopisih vplivajo na kriptograme in na posamezne korake šifrirnega algoritma. Diference, ki jih opazuje ta napad, so modularne razlike in XOR razlike 32-bitnih besed. Pomembno je opaziti, da ti dve diferenci hkrati povesta več, kot vsaka posamezno. V naši

notaciji bomo modularne razlike pisali kot vsoto oz. razliko potenc 2, da je razvidna tudi XOR diferenca. Na primer:

$$X = 01011101$$
$$Y = 11001111$$
$$X - Y = -2 + 2^4 - 2^7$$

Diferenca, ki jo opazujemo, je torej oblike $\Delta X = X' - X$. Za dve k-bločni besedili $M = (M_0, M_1, \ldots, M_{k-1})$, $N = (N_0, N_1, \ldots, N_{k-1})$ je polna diferenca zgoščevalne funkcije zaporedje razlik v izvlečkih blokov:

$$\Delta H_0 \xrightarrow{(M_0,N_0)} \Delta H_1 \xrightarrow{(M_1,N_1)} \Delta H_2 \xrightarrow{(M_2,N_2)} \quad \ldots \quad \Delta H_{k-1} \xrightarrow{(M_{k-1},N_{k-1})} \Delta H.$$

Velja, da je $\Delta H_i = \Delta IV_i$, ker se izvleček enega bloka uporabi kot inicializacijski vektor naslednjega. Ker ja začetni IV_0 za obe besedili enak, je $\Delta H_0 = 0$. Če je končni $\Delta H = 0$, imamo trčenje, diferenco pri kateri se to zgodi, pa imenujemo kolizijska diferenca. Diferenco lahko določimo tudi podrobneje, torej za vsak krog kompresijske funkcije posebej:

$$\Delta H_i \xrightarrow{P_1} \Delta R_{i+1,1} \xrightarrow{P_2} \Delta R_{i+1,2} \xrightarrow{P_3} \Delta R_{i+1,3} \xrightarrow{P_4} \Delta R_{i+1,4} = \Delta H_{i+1},$$

kjer je P_j verjetnost, da je po j-tem krogu kompresijske funkcije diferenca še izpolnjena.

3.2.2 Kolizijska diferenca in pogoji zanjo

Napad poišče trčenje med dvema 1024-bitnima sporočiloma, $M = (M_0, M_1)$ in $N = (N_0, N_1)$. Ciljno kolizijsko diferenco sestavljata dve iteraciji in je oblike:

$$\Delta H_0 \xrightarrow{(M_0,N_0)} \Delta H_1 \xrightarrow{(M_1,N_1)} \Delta H = 0,$$

kjer je:

$$\Delta M_0 = N_0 - M_0 = (0, 0, 0, 0, 2^{31}, 0, 0, 0, 0, 0, 0, 2^{15}, 0, 0, 2^{31}, 0)$$

$$\Delta M_1 = N_1 - M_1 = (0, 0, 0, 0, 2^{31}, 0, 0, 0, 0, 0, 0, -2^{15}, 0, 0, 2^{31}, 0)$$

$$\Delta H_1 = (2^{31}, 2^{31} + 2^{25}, 2^{31} + 2^{25}, 2^{31} + 2^{25}).$$

Za to diferenco so prof. Wang in sodelavci določili karakteristike za vsak korak obeh iteracij kompresijske funkcije in iz njih izpeljali predlagane zadostne pogoje zanje. Ti pogoji določajo vsebino registrov algoritma kompresijske funkcije in vsebine inicializacijskega vektorja za drugi blok. Izpeljava zadostnih pogojev je obsežno in zapleteno sklapljanje pogojev za vsak bit vsakega koraka kompresijske funkcije, kjer je potrebno izpolniti karakteristike. Ker presegajo obseg tega pregleda, si bralec lahko primer izpeljave ogleda v [5], kjer so tudi v celoti tabelirane karakteristike in pogoji, ki so jih izpeljali Wang in sodelavci.

3.2.3 Priredba besedil (message modification)

Besedilo, ki zadošča v prejšnjem podrazdelku omenjenim pogojem, lahko iščemo popolnoma naključno (generirano naključna besedila in jih testiramo), vendar je to preveč neučinkovito, da bi bilo praktično uporabno. Zato je potrebno tako verjetnostno metodo nadgraditi s prijemi, ki nam zagotavljajo izpolnitev vsaj dela pogojev. Izkaže se, da je možno naključno generirano besedilo prirediti, tako da izpolnimo del pogojev za prvo polovico kompresijske funkcije. To dosežemo z naslednjima tehnikama:

Enobesedno Prirejanje (Single-Message Modification) Vse pogoje za prvi krog kompresijske funkcije je možno izpolniti s preprostimi transformacijami posameznih besed besedila. Če bi recimo radi na nekem koraku dosegli, da za register R velja: $R_7 = 0, R_{12} = 0, R_{20} = 0$, kjer je R_i i-ti bit v registru R, potem najprej s kompresijsko funkcijo izračunamo R, s pomočjo neprirejene besede m, in potem izračunamo nova R' in m':

$$R' = R - R_7 \cdot 2^6 - R_{12} \cdot 2^{11} - R_{20} \cdot 2^{19}$$

$$m' = ((R' - R) \ggg 17) + m$$

Če se spomnimo algoritma kompresijske funkcije (shema 2.2), ni težko videti, da sedaj željeni pogoj za R velja.

Če z enobesednimi priredbami izpolnimo pogoje prvega kroga kompresijske funkcije na M_0 in pustimo, da so ostali pogoji izpolnjeni po naključju, dosežemo, da je diferenca za prvi blok izpolnjena z verjetnostjo 2^{-49}. Če enak postopek izvedemo na M_1, je diferenca za drugi blok izpolnjena z verjetnostjo 2^{-37}.

Večbesedno Prirejanje (Multi-Message Modification) Če želimo zagotoviti izpolnjene pogoje iz drugega kroga, podobno kot smo to storili za prvi krog, je potrebno za posamezen bit prirediti več besed besedila, ne le enega. Temu pravimo večbesedno prirejanje in z njim je mogoče izpolniti del pogojev za drugi krog kompresijske funkcije.

Z večbesednim prirejanjem po enobesednem se verjetnost za izpolnitev diference prvega bloka izboljša na 2^{-37}, verjetnost na drugem bloku pa na 2^{-30}.

3.2.4 Shema algoritma in njegova zahtevnost

Iz opisanega lahko povzamemo napad v sledečo psevdokodo:

1. Dokler ne dobiš prvega bloka M_0, ki izpolnjuje pogoje za diferenco, ponavljaj:
   - Izberi naključen blok M_0.
   - Priredi izbrani M_0 s tehnikami, opisanimi v prejšnjem podrazdelku.

- Testiraj, če je diferenca izpolnjena, tako da izračunaš kompresijsko funkcijo na M_0 in N_0.

2. Dokler ne dobiš še drugega bloka in s tem celotnega trčenja ponavljaj:
   - Izberi naključen blok M_1.
   - Priredi izbrani M_1 s tehnikami opisanimi v prejšnjem podrazdelku.
   - Preveri, če besedili $M = (M_0, M_1)$ in $N = (N_0, N_1)$ trčita.

Algoritem lahko implementiramo tako, da pri naključnem iskanju M_0 spreminjamo le zadnji dve besedi, ker lahko na ta način enobesedno prirejanje prvih 14 besed opravimo le enkrat. Tako moramo za vsak novo izbran blok M_0 opraviti le dve enobesedni prirejanji in vsa večbesedna. Te operacije skupaj ne presežejo trajanja dveh izračunov MD5, tako da zahtevnost enega poskusa ne preseže štirih izračunov MD5. Iz verjetnosti uspeha prvega dela algoritma je razvidno, da pričakovana zahtevnost algoritma tako ne preseže 2^{39} izračunov MD5. Iz istih argumentov sledi, da zahtevnost drugega dela algoritma ne preseže 2^{32} izračunov MD5.

3.3 Dopolnitev potrebnih pogojev

Kasneje leta 2005 sta J. Liang in X. Lai s protiprimeri pokazala, da pogoji, ki so jih predlagali Wang in sodelavci, niso zadostni ter, da se da določene njihove pogoje omiliti. Predlagala sta nov nabor pogojev, za katere se domneva, da so pravilni, vendar to še ni dokazano. Predstavila sta tudi novo tehniko za izpolnjevanje pogojev za diference, imenovano small range searching, ki lahko izpolni še 7 dodatnih pogojev, ki jih prirejanje besedil ne more. V opis te metode se ne bomo spuščali, ker gre za pristop, ki je podoben prirejanju besedil. Zainteresiran bralec lahko podrobnosti, kot tudi seznam novih pogojev, najde v [6]. Z novimi pogoji in metodo izpolnjevanja se hitrost napada izboljša na 2^{34} oz. 2^{28} izračunov MD5 za prvi oz. drugi blok.

3.4 Tuneliranje

Do naslednjega pomembnejšega teoretičnega razvoja je prišlo leta 2006, ko je profesor Vlastimil Klima predstavil novo tehniko, imenovano tuneliranje, ki pomaga zadostiti pogojem diference. Z vsemi dotedanjimi pristopi lahko izpolnimo del pogojev za diferenco, potem pa pridemo do točke preverjanja (TP; angl. point of verification), ko nam preostane le še, da preverimo preostale pogoje, ki so izpolnjeni le po naključju. Čeprav so možne implementacije, ki generirajo nova naključna besedila tako, da zadržijo del že izpolnjenih pogojev (glej razdelek o osnovnem diferenčnem napadu; odstavek o zahtevnosti algoritma), je še vedno potrebno vsakič znova opraviti del prirejanja, da zopet dosežemo TP. Tuneliranje je pristop, ki ga uporabimo v TP, da generiramo nova besedila, ki so kandidati za trčenje, in ob tem zadržimo (z veliko verjetnostjo) vse pogoje, ki smo jih izpolnili s prirejanjem besedil.

3.4.1 Tuneli

Tuneli so skupine bitov v prvih 16 registrih kompresijske funkcije, ki jih lahko pod določenimi pogoji spreminjamo in pri tem zadržimo že izpolnjene pogoje. Spomnimo se, da je spreminjanje vrednosti teh registrov preprosto, saj je ekvivalentno enobesednemu prirejanju besedil iz osnovnega diferenčnega napada.

Od splošnih lastnosti tunelov sta najpomembnejši naslednji:

Moč je ustrezna številu kandidatov za trčenje, ki jih lahko ustvarimo s tem tunelom. Če z n označimo moč tunela, lahko z njim zgeneriramo 2^n kandidatov za trčenje. Nekateri tuneli imajo posebne dodatne pogoje s katerimi jim lahko zagotovimo maksimalno moč. Ti pogoji niso potrebni za trčenje samo, omogočajo le večje število generiranih kandidatov.

Če imamo da tunela, prvega z močjo i in drugega z j, imata skupaj moč $i + j$, ker so tuneli med seboj neodvisni.

Cena nam pove povprečno potrebno število izračunov korakov kompresijske funkcije, da s tunelom zgeneriramo novega kandidata za trčenje.

Poznamo več vrst tunelov:

Deterministični Vsaka sprememba bitov tunela ne more pokvariti že izpolnjenih pogojev in tako da novega kandidata za trčenje z verjetnostjo 1.

Probablistični Obstaja možnost, da prirejanje bitov tunela pokvari določene že izpolnjene pogoje. Potrebno je dodatno testiranje teh pogojev. Ker tunel zgenerira novega kandidata le z določeno verjetnostjo, ki je manjša od 1, je kot moč tunela definirana pričakovana moč.

Dinamični Pri dinamičnih tunelih je to, katere bite lahko spremenimo, odvisno od ustreznih bitov v drugih registrih. Da polno izkoristimo moč takih tunelov, si moramo ustrezno nastavljati tudi te pogojne bite.

Iskanje tunelov in analiza njihovih lastnosti je zelo netrivialen postopek, ki zahteva natačno poznavanje lastnosti kompresijske funkcije. Klima je v svojem članku [7] predstavil šest tunelov s skupno močjo približno 27. Ker je tunele iskal na podlagi diference in pogojev iz [6] to pomeni, da za prvi blok pričakovano potrebnih le 2^7 točk preverjanja, za drugi blok pa celo samo dve. Napad sam je v osnovi enak kot v [6], le da za vsako doseženo točko preverjanja v več vgnezdenih zankah preverimo vse možne kombinacije vrednosti tunelov.

Klima v svojem članku ne poda analitične obravnave zahtevnosti algoritma s tuneli. Eksperimenti kažejo, da je zahtevnost ekvivalentna približno $\frac{2^{27.6}}{3}$ izračunov MD5, vendar gre za precej grobo oceno. Napad povprečno poišče trčenje za MD5 v približno minuti na navadnem prenosnem računalniku. Velja omeniti, da je Klima tuneliranje umetno dodal v diferenčno shemo iz [6]. Klima sam opozarja, da bi, če bi razvil ustrezno diferenčno shemo, ki je namenjena uporabi s tuneli, lahko v njej izkoristil večje število tunelov in tako dosegel še večjo moč napada.

Poglavje 4

Zaključek

V seminarski nalogi je opisana vrsta modernih prijemov, ki sestavljajo novejše napade na zgoščevalne funkcije. Ker gre za splošne metode in ker so mnoge zgoščevalne funkcije zgrajene na istih principih, so isti prijemi uporabni še na vrsti drugih zgoščevalnih funkcij, recimo RIPEMD, HAVAL, družini SHA. To sicer ne pomeni, da so vse že razbite, saj je aplikacija teh prijemov (konstrukcija diferenc, iskanje tunelov...) dolga in zahtevna. Vseeno pa lahko z gotovostjo trdimo, da je doba te generacije zgoščevalnih funkcij že skoraj minila. Zadnji varnejši predstavniki so močnejši člani družine SHA, recimo SHA-2, vendar je zaradi algoritmičnih podobnosti tudi to najbrž le vprašanje časa. Zaradi tega, je trenutno v teku javni razpis za standard SHA-3, ki se konča oktobra 2008. Leta 2012 naj bi izmed oddanih predlogov izbrali zgoščevalno funkcijo, ki bo postala nov standard.

Literatura

[1] Ron Rivest. *The MD5 Message-Digest Algorithm, Request for Comments:1321*, Dostopno na URL: http://rfc.net/rfc1321.html, 1992.

[2] B. Boer, A. Bosselaers. *Collisions for the compression function of MD5*, Advances in Cryptology, Eurocrypt'93 Proceedings, Springer-Verlag, 1994.

[3] H. Dobbertin. *The status of MD5 after a recent attack*, CryptoBytes 2 (2), 1996, Dostopno na URL: ftp://ftp.rsasecurity.com/pub/cryptobytes/crypto2n2.pdf.

[4] X. Wang et al. *Collisions for hash functions MD4, MD5, HAVAL-128 and RIPEMD*, rump session of Crypto'04, E-print, 2004.

[5] X. Wang, H. Yu. *How to break MD5 and other hash functions*, predtavljeno na EUROCRYPT 2005, Dostopno na URL: http://merlot.usc.edu/cs531-f07/papers/Wang05a.pdf.

[6] J. Liang, X. Lai. *Improved Collision Attack on Hash Function MD5*, Cryptology ePrint Archive: Report 425/2005, 2005, Dostopno na URL: http://eprint.iacr.org/2005/425.pdf.

[7] V. Klima. *Tunnels in Hash Functions: MD5 Collisions Within a Minute*, Cryptology ePrint Archive: Report 105/2006, 2006, Dostopno na URL: http://eprint.iacr.org/2006/105.