HRVATSKA AKADEMSKA I ISTRAŽIVAČKA MREŽA
CROATIAN ACADEMIC AND RESEARCH NETWORK

Algoritmi za izračunavanje sažetka

CCERT-PUBDOC-2006-08-166

CARNet CERT u suradnji s LS&S

Sigurnosni problemi u računalnim programima i operativnim sustavima područje je na kojem CARNet CERT kontinuirano radi.

Rezultat toga rada ovaj je dokument, koji je nastao suradnjom CARNet CERT-a i LS&S-a, a za koji se nadamo se da će Vam koristiti u poboljšanju sigurnosti Vašeg sustava.

CARNet CERT, www.cert.hr - nacionalno središte za **sigurnost računalnih mreža** i sustava.

LS&S, www.lss.hr - laboratorij za sustave i signale pri Zavodu za elektroničke sustave i obradbu informacija Fakulteta elektrotehnike i računarstva Sveučilišta u Zagrebu.

Sadržaj

1. Uvod

Tema ovog dokumenta su algoritmi za izračunavanje sažetka (eng. *hash*) i njihova primjena. Izračunavanje sažetka u principu predstavlja kreiranje digitalnog otiska fiksne veličine za ulazne podatke. Digitalni otisak, koji algoritam daje kao rezultat, je obični kratki pseudo-slučajni niz znakova - binarni podaci zapisani u heksadecimalnoj notaciji.

Ukoliko je algoritam dobar, onda se za svaki različiti ulazni niz podataka algoritmom dobiva različiti sažetak, tj. dobiveni digitalni otisak je jedinstven i ne može se dobiti pomoću neke druge ulazne vrijednosti. Nažalost, zbog činjenice da se s manjim brojem znakova treba prezentirati veći broj znakova, jedinstvenost dobivenog sažetka je teško postići. I ukoliko napadač otkrije mogućnost falsificiranja ulaznog niza podataka, on može krivotvoriti ulazne nizove podataka. Postojeći algoritmi za izračunavanje sažetka u većoj ili manjoj mjeri mogu postići jedinstvenost sažetka, gdje se mjera kvalitete izražava u broju kolizija sažetaka (eng. *hash collisions*). Zbog tog svojstva jedinstvenosti, sažetak se može primijeniti i primjenjuje se u svrhe kriptografije, autentikacije i identifikacije.

U ovom dokumentu opisane su metode koje se koriste u izračunu sažetka, najčešće korišteni algoritmi za izračunavanje te njihova primjena.

2. Osnovni principi

Algoritmima za izračun sažetka se smatraju oni koji zadovoljavaju minimalno dva zahtjeva:

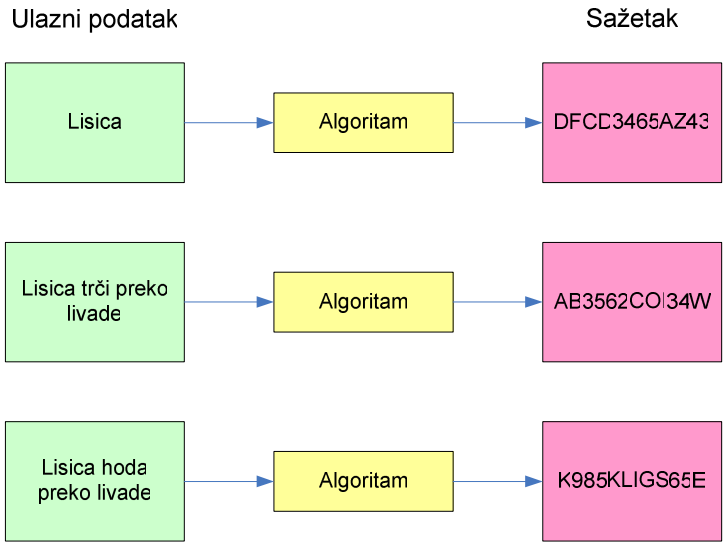
- kompresija prema kojoj se ulazni podaci nedefinirane veličine sažimaju u rezultat fiksne veličine, i
- jednostavnost izračuna.

Osnovno svojstvo svih efikasnih algoritama za izračunavanje sažetka je zahtjev prema kojemu se za dva različita sažetka izračunata istim algoritmom, moraju i ulazni podaci iz kojih su sažeci izračunati, biti različiti. To znači da algoritmi za izračunavanje sažetka moraju biti deterministički.

Nasuprot prethodnom svojstvu, drugo svojstvo definira kako algoritmi za izračunavanje sažetka ne moraju biti nužno injektivni, tj. jednakost dva sažetka ne mora garantirati da su ulazni podaci iz kojih su oni nastali također jednaki. Ako se izračuna sažetak za jednu vrijednost i nakon toga se promijeni samo jedan bit u toj vrijednosti, novi sažetak bi trebao biti potpuno različit od prethodnog.

Tipičan algoritam za izračunavanje sažetka može prihvatiti bilo koju veličinu ulaznog podatka, a kao izlaz obično daje niz bitova fiksne dužine, ali isti može biti i varijabilne dužine. Neki algoritmi za izračunavanje sažetka će za istu dužinu ulaznih podataka uvijek dati jednaku dužinu izlaznog podatka – takvi algoritmi se nazivaju permutacije.

Princip rada algoritma za izračunavanje sažetka prikazan je na sljedećoj slici.



Slika 1: Princip rada algoritma za izračunavanje sažetka

2.1. Područja primjene

Algoritmi za izračunavanje sažetka imaju dva glavna područja primjene:

- kriptografija / zaštita podataka i
- detekcija pogreške / provjera ispravnosti podataka.

Algoritmi koji se primjenjuju u ova dva područja su vrlo različiti. Za algoritme koji se primjenjuju u kriptografiji glavno svojstvo je što su jednosmjerni, tj. njihovim korištenjem se iz poznavanja sažetka ne može matematičkim operacijama doći do originalnog podatka iz kojeg je sažetak izračunat. Kod kreiranja takvih algoritama osnovna pretpostavka je da postoji zlonamjerni korisnik čija je glavna namjera naći podatak čiji će sažetak biti jednak nekom pronađenom sažetku kako bi taj podatak iskoristio za npr. lažiranje autentikacije.

Algoritmi koji se primjenjuju za detekciju pogreške i provjeru ispravnosti podataka koriste se za detekciju pogreške prilikom prijenosa podataka komunikacijskim kanalom ili za detekciju pogreške prilikom pohranjivanja podataka na medije. Takvi algoritmi koriste se u proceduri koja se naziva ciklička provjera redundancije (eng. CRC - *Cycliy Redundancy Check*) koja je u raznim implementacijama danas u vrlo širokoj upotrebi. Kod takvih algoritama nije bitno jesu li oni otporni na napade malicioznih korisnika te se ne smatra nedostatkom algoritma ako je moguće iz dvije različite poruke generirati isti sažetak.

Primjena algoritama za izračunavanje sažetka pronalazi se i u sljedećim sigurnosnim protokolima:

- X.509 – standard za infrastrukturu javnih ključeva (eng. PKI - *Public Key Infrastructure*), definira standardne formate za certifikate javnih ključeva i algoritam verifikacije certifikata,
- SSL (eng. *Secure Socket Layer*) i njegova nadogradnja TLS (eng. *Transport Layer Security*) koji omogućavaju autentikaciju i zaštitu transportnog puta,
- PGP (eng. *Pretty Good Privacy*) za kriptiranje i autenticiranje podataka,
- S/MIME (eng. *Secure / Multiple Internet Mail Extension*) za osiguranje integriteta poruka,
- IPSec (eng. *IP Security*) koji predstavlja skup protokola za sigurnu razmjenu IP paketa, itd.

2.2. Svojstva kriptografskih algoritama

Kriptografski algoritmi za izračunavanje sažetka su specifični algoritmi koji su modificirani kako bi im se podigla razina zaštite od napada. Takvi algoritmi morali bi biti što sličniji funkciji generiranja slučajnih brojeva, ali istovremeno moraju imati deterministički karakter i biti dovoljno efikasni kako izračun ne bi trajao dugo.

Kriptografski algoritam za izračunavanje sažetka se smatra nesigurnim ako je moguće:

- nalaženje prethodno nepoznatog podatka za koji algoritam kao rezultat daje traženi sažetak, ili
- nalaženje „kolizija“, tj. dva različita podatka koji za rezultat daju isti sažetak.

Napadač koji može napraviti bilo koju od prethodno navedenih stvari može to iskoristiti za npr. neovlaštenu autorizaciju korištenjem zamjenskog podatka. Iako su gore navedeni uvjeti minimum, idealno bi bilo kad ne bi bilo moguće naći niti dva različita ulazna podatka koji daju sličan (ne isti) sažetak i kada se iz samog sažetka ne bi moglo baš ništa saznati o samom algoritmu i početnom podatku.

Ne postoji formalna definicija osnovnih svojstava koja bi definirala kriptografski algoritam za izračunavanje sažetka, ali sljedeća svojstva se uglavnom smatraju adekvatnim preduvjetima:

- jednosmjernost (eng. *preimage resistance*) – uz poznatu vrijednost *h* mora biti TEŠKO pronaći *m* takav da je *h = hash(m)*, gdje *hash* predstavlja algoritam, a pojam TEŠKO je matematički definiran,
- jednoznačnost ili slaba otpornost na koliziju (eng. *2nd-preimage resistance*) – uz poznatu vrijednost *m1* mora biti TEŠKO pronaći vrijednost *m2* (različitu od *m1*) takvu da je *hash(m1) = hash(m2)*, gdje *hash* predstavlja algoritam, a pojam TEŠKO je matematički definiran,
- općenita jednoznačnost ili jaka otpornost na koliziju (eng. *collision resistance*) – mora biti TEŠKO pronaći bilo koje dvije različite vrijednosti *m1* i *m2* za koje vrijedi *hash(m1) = hash(m2)*, gdje *hash* predstavlja algoritam, a pojam TEŠKO je matematički definiran.

Algoritam koji zadovoljava gore navedena svojstva svejedno može imati neka neželjena svojstva. Na primjer većina popularnih algoritama nije otporna na napad produženjem ulaznog podatka – uz poznati *hash(m1)* i dužinu podatka *m1* i nepoznat podatak *m1* korištenjem prikladnog *m2* napadač može izračunati *hash(m1 ‖ m2)* (gdje ‖ označava konkatenaciju) što se može iskoristiti za probijanje određenih autentikacijskih shema koje su zasnovane na *hash* funkcijama..

Idealni algoritam za izračunavanje sažetka je maksimalno „dosadan“ – nema zanimljivih svojstava poput produživanja dužine i jedina razlika između algoritma i funkcije generiranja slučajnih brojeva je ta što je algoritam determinističkog karaktera i što je efikasan za računanje.

2.3. Osnovna klasifikacija

Algoritmi za izračunavanje sažetka dijele se u dvije osnovne skupine:

- algoritmi za izračunavanje sažetka bez ključa (eng. MDC - *Modification Detection Codes*) koji za ulaz koriste samo jedan parametar (ulazna poruka) i
- algoritmi za izračunavanje sažetka s ključem (eng. MAC – *Message Authentication Codes*) koji za ulaz koriste dva parametra (ulazna poruka i ključ).

Svrha algoritama za izračunavanje sažetka bez ključa (MDC) je izrada sažetka kojima se osigurava integritet podataka koji je zahtjevan od različitih aplikacija. MDC-ovi se mogu dalje podijeliti na sljedeće podvrste:

- jednosmjerne *hash* funkcije (eng. OWHF – *One-Way Hash Functions*) i
- *hash* funkcije otporne na koliziju (eng. CRHF – *Collision Resistant Hash Functions*).

Zahtjevi koji se stavljaju pred jednosmjerne *hash* funkcije (OWHF) su svojstva jednosmjernosti i jednoznačnosti (slaba otpornost na koliziju). *Hash* funkcije otporne na koliziju (CRHF) moraju zadovoljavati svojstva jake i slabe otpornosti na koliziju. Također, *hash* funkcije otporne na koliziju uobičajeno podržavaju i zahtjev jednosmjernosti, ali to nije zahtijevano. Uobičajeno se u literaturi OWHF nazivaju slabima jednosmjernim *hash* funkcijama, a CRHF se nazivaju jakim jednosmjernim *hash* funkcijama.

Pred algoritme za izračun sažetaka bez ključa se u praksi mogu staviti i dodatni zahtjevi:

- međusobna nepovezanost (eng. *non-correlation*) ulaznih i izlaznih podataka – između ulaznih i izlaznih bitova ne smije postojati korelacija,
- otpornost na blisku koliziju (eng. *near-collision resistance*) – mora biti TEŠKO pronaći bilo koje dvije različite vrijednosti *m1* i *m2* za koje vrijedi da se za njih izračunati sažetci *hash(m1)* i *hash(m2)* razlikuju u MALO bitova,
- lokalna jednosmjernost (eng. *local one-wayness*) - mora biti TEŠKO pronaći bilo koji podniz ulaznog niza, kao i sam ulazni niz, a čak i kad je poznat dio ulaznog niza, mora biti TEŠKO pronaći ostatak.

Zahtjevi koji se stavljaju pred algoritme za izračun sažetka s ključem (MAC) su kompresija podataka i jednostavnost izračuna. Također, dodatni zahtjev je i zahtjev računske otpornosti (eng. *computation resistance*) koji definira da niti za jedan ili više poznatih parova ulaznih podataka i izračunatog sažetka, računski nije moguće izračunati neki drugi par ulaznog podatka i sažetka koji je različit od poznatih. U slučaju kada računska otpornost nije ispunjena, MAC algoritam je podložan krivotvorenju. I dok ispunjen zahtjev računske otpornosti implicira ispunjenje zahtjeva ne-otkrivanja ključa (eng. *key non-recovery*), obrat ne vrijedi jer nije nužno otkriti ključ da bi se kreirali novi parovi ulazne poruke i s njom povezanog sažetka.

3. Metodologija dizajna algoritama za izračun sažetka

Postoji nekoliko osnovnih metodologija dizajna algoritama za izračunavanje sažetka:

- dizajn baziran na blok metodi enkripcije,
- dizajn baziran na modularnoj aritmetici i
- ekskluzivan dizajn za izračunavanje sažetka.

Algoritmi bazirani na blok metodi enkripcije baziraju se na višestrukoj upotrebi i kombinaciji već poznatih enkripcijskih funkcija koje se baziraju na blok metodi enkripcije. Snaga i efikasnost takvih algoritama ovise o primijenjenoj blok metodi enkripcije i o kompleksnosti logičkih i iterativnih operacija kojima se te funkcije kombiniraju u dizajnu algoritma.

Algoritmi bazirani na modularnoj aritmetici nastali su iz potrebe za pojeftinjenjem implementacije algoritma za izračunavanje sažetka. Međutim, algoritmi tog dizajna nisu se pokazali kao vrlo uspješni iz perspektive sigurnosti pa danas u upotrebi ne postoje algoritmi za izračunavanje sažetka bazirani na modularnoj aritmetici. Iz tog razloga oni nisu detaljno obrađeni u ovom dokumentu.

Algoritmi koji su posebno dizajnirani za ostvarenje funkcionalnosti izračunavanja sažetka nastali su iz potrebe za povećanom sigurnošću. Njihov dizajn također se bazira na višestrukoj upotrebi i kombinaciji enkripcijskih funkcija. Pri tome enkripcijske funkcije mogu biti bazirane na blok metodi enkripcije, ali i ne moraju, a glavna snaga tih algoritama je njihov efikasan i teže probojan dizajn koji se obično bazira na nekoj teorijski dobro ispitanoj logičkoj konstrukciji. Primjer dizajna takvog algoritma je Merkle-Damgard konstrukcija na kojoj su bazirani MD5 i SHA-1 algoritmi za izračunavanje sažetka.

3.1. Blok metoda enkripcije

Blok enkripcija (eng. *block cipher*) je simetrična enkripcija koja obrađuje grupe bitova fiksne dužine (blokove) pomoću nevarijabilne transformacije. Kod takve enkripcije ulaz je npr. 128-bitni blok tekstualnog formata, a izlaz je odgovarajući 128-bitni enkriptirani blok teksta. Transformacija je kontrolirana pomoću drugog ulaza – tajnog ključa. Dekripcija je slična – algoritam uzima 128-bitni blok kriptiranog teksta i tajni ključ te na izlazu daje originalni 128-bitni blok teksta.

Slika 2: Blok enkripcija / dekripcija

Za enkripcije tekstova većih od 128 bita koriste se različite metode rastavljanja podataka na manje blokove koji se onda enkriptiraju raznim metodama uz ugrađivanje većeg ili manjeg nivoa međusobne ovisnosti među blokovima. Postojeće metode su ECB (eng. *Electronic Codebook*), CBC (eng. *Cipher Block Chaining*), OFB (eng. *Output Feedback*) s manjim nivoom ovisnosti i CCM (eng. *Counter with CBC-MAC*), EAX (*Conventional Authenticated-Encryption Mode*) i OCB (eng. *Offset Codebook mode*) s većim nivoom ovisnosti i zaštite, ali one nisu tema ovog dokumenta.
Alternativa blok enkripciji je slijedna (eng. *stream*) enkripcija koja obrađuje bit po bit ulaznog podatka pri čemu se transformacija mijenja tijekom procesa enkripcije. Razlike između blok i slijedne enkripcije nije uvijek jasna jer blok enkripcija uz korištenje nekih metoda za enkripciju većih blokova podataka postaje vrlo slična slijednoj enkripciji.
Blok enkripcija se bazira na 2 uparena algoritma – jedan za enkripciju E, a drugi za dekripciju E^{-1}. Oba algoritma uzimaju 2 ulaza: ulazni blok podataka od n bita i ključ od k bitova, te na izlazu daju blok od n bitova. Ako algoritmi koriste isti ključ, dekripcija je inverzna funkcija enkripcije gdje za svaku ulazni blok M i ključ K vrijedi sljedeće:

$$E_k^{-1}(E_k(M)) = M$$

Za svaki ključ, funkcija E_K je permutacija (bijektivno preslikavanje) na skupu ulaznih blokova. Svaki ključ služi za izbor jedne od mogućih 2^n permutacija gdje n predstavlja broj bitova ulaznog bloka.
Veličina ulaznog bloka n je obično 64 ili 128 bita iako neke vrste blok enkripcije podržavaju i varijabilne dužine blokova. 64 bita je bila uobičajena dužina do sredine 90-ih godina kada se zbog potrebe za povećanjem sigurnosti prešlo na 128-bitne i veće blokove.
Većina blok enkripcija bazira se na uzastopnoj upotrebi jednostavnije funkcije i takav se dizajn naziva iterativna blok enkripcija, a uobičajeni broj iteracija je između 4 i 32. Struktura takvih enkripcija u većini slučajeva odgovara strukturi Feistel mreže u kojoj se kao komponente kombiniraju različite aritmetičke i logičke (posebno XOR) operacije, te permutacije.
Jedna od najstarijih i najutjecajnijih blok enkripcija je DES (eng. *Data Encryption Standard*) enkripcija koju je razvio i objavio IBM 1977. godine, a koja je 2001. godine dobila kao nasljednika AES (eng. *Advanced Encryption Standard*) enkripciju. DES enkripcija se danas smatra nedovoljno sigurnom za većinu aplikacija zbog premale dužine ključa od 56 bita – poznati su slučajevi kada su DES ključevi bili probijeni unutar 24 sata. Također, dokazane su i neke teoretske slabosti algoritma, iako ih je u praksi nemoguće implementirati. Zbog toga je razvijena nadogradnja algoritma u obliku trostruke DES (eng. *Triple DES*) enkripcije koja daje povećanu razinu sigurnost, iako su i za tu enkripciju otkriveni teoretski napadi pa se svi oblici DES enkripcije postepeno zamjenjuju sa svojim nasljednikom AES enkripcijom.

3.1.1. Slijedna metoda enkripcije

Slijedna metoda enkripcije je vrsta simetrične enkripcije koja se ponekad upotrebljava umjesto blok metode enkripcije zbog toga što radi znatno brže. Slijedna enkripcija za razliku od blok metode enkripcije ne uzima blok podataka kao ulaz već je ulaz obično samo jedan bit podatka. Kod blok metode enkripcije za isti ulazni podatak, uz upotrebu istog ključa, rezultat će uvijek biti isti sažetak. Kod slijedne enkripcije to nije slučaj.

Slijedna enkripcija generira tzv. „*keystream*“, tj. niz bitova koji predstavljaju ključ, a enkripcija se obavlja kombiniranjem ključa s ulaznim podatkom i to obično XOR logičkom operacijom. Generiranje ključa može biti:

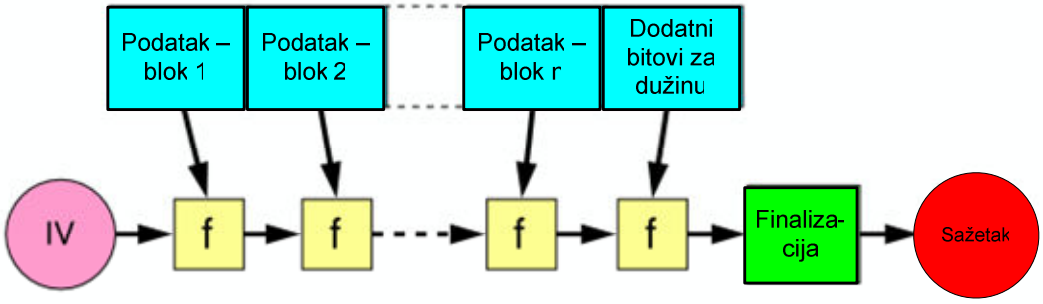
- neovisno o ulaznom podatku i sažetku i tada se enkripcija naziva sinkronom slijednom enkripcijom ILI
- ovisno o ulaznom podatku i sažetku i tada se radi o samo-sinkronizirajućoj slijednoj enkripciji.

Većina slijednih enkripcija je sinkrona. Najpoznatiji algoritam baziran na slijednoj metodi enkripcije je RC4 algoritam koji se primjenjuje kod SSL i WEP (eng. *Wired Equivalent Privacy*) protokola.

3.2. Merkle-Damgard konstrukcija

Kriptografski algoritam za izračunavanje sažetka mora omogućiti konverziju ulaznih podataka različite dužine u izlazni podatak fiksne dužine. To se može postići razlamanjem ulaznog podatka u seriju blokova iste dužine koji se potom obrađuju istom kompresijskom funkcijom. Generički princip rada takvih algoritama opisali su Ralph Merkle i Ivan Damgard pa se taj generički princip rada naziva Merkle-Damgard konstrukcija.

Kompresijska funkcija u takvom algoritmu može biti posebno dizajnirana ili se može bazirati na blok metodi enkripcije. Algoritam je u tom slučaju otporan na kolizije rezultata onoliko koliko je na njih otporna korištena kompresijska funkcija i u njoj se nalazi izvorište svih kolizija. Na Merkle-Damgard konstrukciji bazirana je većina danas korištenih algoritama za izračunavanje sažetka uključujući MD5 i SHA-1 algoritme.



Slika 3: Merkle-Damgard konstrukcija

Princip Merkle-Damgard konstrukcije prikazan je na gornjoj slici gdje *f* predstavlja kompresijsku funkciju. Algoritam započinje s inicijalnom vrijednošću ili incijalizacijskim vektorom *IV* koji predstavlja fiksnu vrijednost koja je ovisna o algoritmu ili implementaciji. Za svaki blok ulaznih podatka, kompresijska funkcija uzima rezultat prethodne kompresijske funkcije (prva funkcija uzima inicijalizacijski vektor), kombinira ga s blokom ulaznih podataka i na osnovu toga izračunava rezultat koji se prosljeđuje sljedećoj kompresijskoj funkciji. Zadnji blok ulaznih podataka se nadopunjava 0-bitovima i bitovima koji pohranjuju podatak o dužini cjelokupnog ulaznog podatka kako bi se dobio blok fiksne dužine jednake dužinama prethodnih blokova.

Kako bi se krajnji rezultat dodatno zaštitio, on se obrađuje finalizacijskom funkcijom koja može imati funkciju komprimiranja zadnjeg rezultata u sažetak manje dužine, a također se koristi kao garancija boljeg miješanja podatka i dodavanja efekta lavine (mala promjena na ulazu uzrokuje veliku promjenu na izlazu) na bitove sažetka. Također, finalizacijska funkcija je često bazirana na kompresijskoj funkciji.

Za popularnost ove funkcije zaslužna je činjenica koju su dokazali Merkle i Damgard, a koja definira pravilo prema kojem za postojanje kompresijske funkcije otporne na kolizije i algoritam dobiven konstrukcijom pomoću te kompresijske funkcije mora biti otporan na kolizije.

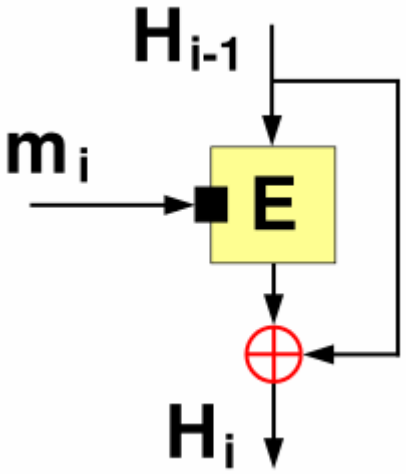
Međutim, Merkle-Damgard konstrukcija ima i neke nedostatke:

- ako je pronađena jedna kolizija produženjem odgovarajućeg ulaznog podatka, vrlo lako ih se može naći još,
- napadi na jednoznačnost funkcije su mnogo efikasniji nego napadi uzastopnim pokušajima,
- višestruke kolizije (više poruka s istim sažetkom) mogu se pronaći vrlo lako nakon pronalaska prve.

Neke od kompresijskih funkcija koje se obično upotrebljavaju u sklopu Merkle-Damgard konstrukcije opisane su u nastavku dokumenta.

3.2.1. Davies-Meyer kompresijska funkcija

Davies-Meyer kompresijska funkcija uzima svaki blok ulaznog podatka m_i kao ključ za enkripciju blok metodom. Kao ulaz enkripcije se koristi rezultat prethodne kompresijske funkcije H_{i-1} u tekstualnom obliku. Rezultat enkripcije se obrađuje XOR funkcijom s rezultatom prethodne kompresijske funkcije, a rezultat toga predstavlja rezultat kompresijske funkcije H_i.
Veličina bloka ulaznog podatka određena je traženom duljinom ključa potrebnog za enkripciju. Ako enkripcija zahtijeva 128-bitni ključ onda će ulazni podatak bit razložen na 128-bitne segmente, a izlaz kompresijske funkcije će također biti dužine 128 bita.
Princip rada Davies-Meyer kompresijske funkcije prikazan je na sljedećoj slici.

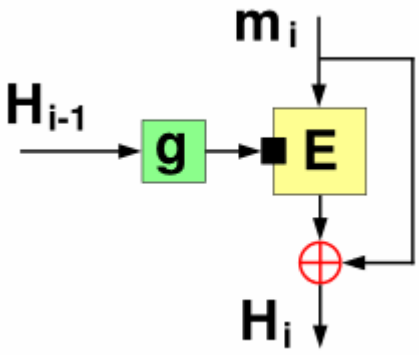


Slika 4: Davies-Meyer kompresijska funkcija

Postoje i varijacije ove kompresijske funkcije koje umjesto XOR operacije koriste neku drugu grupnu operaciju, npr. dodavanjem 32-bitnih cjelobrojnih vrijednosti.

3.2.2. Matyas-Meyer-Oseas kompresijska funkcija

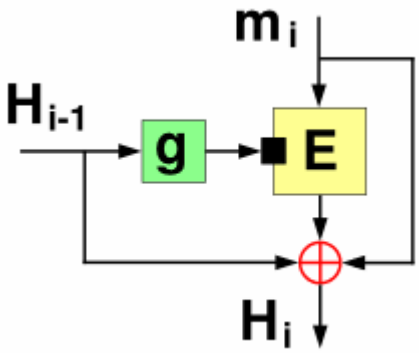
Ova kompresijska funkcija može se smatrati suprotnom Davies-Meyer funkciji. Ovdje se svaki blok ulaznog podatka m_i u tekstualnom formatu kriptira blok metodom gdje se kao ključ koristi rezultat prethodne kompresijske funkcije H_{i-1}. Izlaz enkripcije se obrađuje XOR funkcijom s nekriptiranim inicijalnim blokom ulaznog podatka, a izlaz te funkcije predstavlja i krajnji rezultat kompresijske funkcije H_i.
Ukoliko enkripcijska funkcija zahtijeva različite dužine ulaznog bloka podatka i ključa za kriptiranje, onda će njen rezultat biti nepogodne dužine za sljedeći stupanj gdje se treba koristiti kao ključ za enkripciju. Također enkripcijska funkcija može postavljati i dodatne zahtjeve na ključ. Zbog toga se često implementira dodatna funkcija g koja konvertira izlaz kompresijske funkcije u format pogodan za sljedeći stupanj. Princip rada ove kompresijske funkcije prikazan je na sljedećoj slici.



Slika 5: Matyas-Meyer-Oseas kompresijska funkcija

3.2.3. Miyaguchi-Preneel kompresijska funkcija

Ova funkcija je nadogradnja na Matyas-Meyer-Oseas funkciju s razlikom što se kao ulaz u XOR funkciju na izlazu kompresijske funkcije koristi nekriptirani blok ulaznog podatka m_i, izlaz enkripcijske funkcije i rezultat prethodne kompresijske funkcije H_{i-1}. Princip rada ove funkcije prikazan je na sljedećoj slici.



Slika 6: Princip rada Miyaguchi-Preneel kompresijske funkcije

4. Implementacija algoritama za izračunavanje sažetka

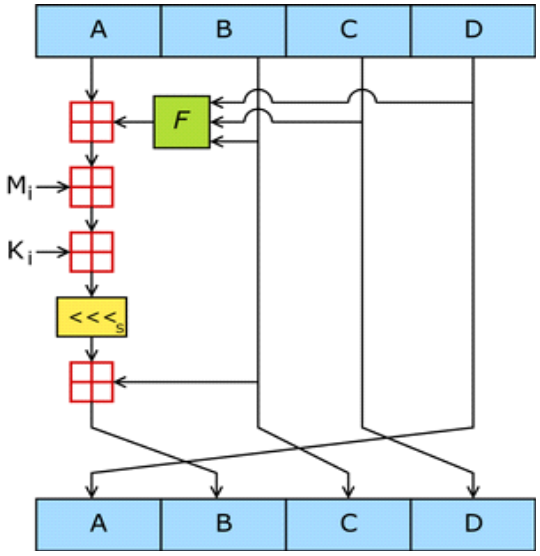
Na principima i dizajnu opisanima u prethodnom poglavlju bazira se velika većina danas korištenih algoritama za izračunavanje sažetka. Poglavlja koja slijede daju opise danas najpopularnijih i najčešće korištenih algoritama.

4.1. MD5 algoritam

MD5 algoritam (eng. *Message-Digest algorithm 5*) je vrlo popularan kriptografski algoritam za izračunavanje sažetka koji koristi 128-bitni sažetak. MD5 je definiran RFC 1321 standardom i koristi se u velikom broju sigurnosnih aplikacija, uglavnom za provjeru integriteta datoteka.

MD5 je razvio 1991. godine Ronald Rivest kao nasljednika algoritma MD4, a već 1996. godine pronađena je ranjivost algoritma koja, iako nije predstavljala kritičnu opasnost po algoritam, navela je stručnjake da preporučuju upotrebu drugih algoritama kao što je SHA-1. Unatoč tome, MD5 je ostao u širokoj upotrebi do 2004. godine kada su otkrivene ozbiljne sigurnosne mane algoritma koje njegovu buduću upotrebu čine upitnom.

Princip rada MD5 algoritma prikazan je na sljedećoj slici.



A, B, C, D – 32 bitne riječi koje opisuju stanje algoritma

F – nelinearna funkcija

M_i – 512-bitni blok ulazne poruke

K_i – 32-bitna konstanta definirana za svaku operaciju

$<<<_s$ – rotacija u lijevo za s mjesta, s definiran za svaku operaciju

+ modulo 2^{32}

Slika 7: Princip rada MD5 algoritma

MD5 algoritam uzima kao ulaz podatak varijabilne dužine koji se kriptira u sažetak dužine 128 bita. Ulazni podatak se razdvaja u blokove od 512 bita, zadnji blok se po potrebi nadopunjava do dužine 512 bita. Nadopunjavanje se radi po sljedećem pravilu – prvo se dodaje bit vrijednosti 1, zatim bitovi vrijednost 0 sve dok duljina poruke nije 64 bita manja od 512 bita, a zadnjih 64 bita sadrži vrijednost koja označava dužinu ulaznog podatka.
Glavni MD5 algoritam radi s 128-bitnim stanjem podijeljenim u četiri 32-bitne riječi označene s A, B, C, D koje su inicijalizirane na fiksne vrijednosti. Algoritam obrađuje jedan po jedan blok ulazne poruke gdje svaki blok modificira stanje algoritma. Obrada jednog bloka sastoji se od 4 stupnja gdje svaki stupanj sadrži 16 operacija baziranih na nelinearnoj funkciji F, modularnom zbrajanju i rotaciji u lijevo. Na gornjoj slici prikazana je jedna operacija unutar jednog stupnja. Funkcija F može biti jedna od sljedeće 4 definicije:

$$F(X,Y,Z) = (X \text{ AND } Y) \text{ OR } ((\text{not } X) \text{ AND } Z)$$
$$G(X,Y,Z) = (X \text{ AND } Z) \text{ OR } (Y \text{ AND } (\text{not } Z))$$
$$H(X,Y,Z) = X \text{ XOR } Y \text{ xOR } Z$$
$$I(X,Y,Z) = Y \text{ xOR } (X \text{ OR } (\text{not } Z))$$

Primjer MD5 sažetka:

```
MD5("Ovo je primjer teksta za dobivanje hash-a.")
  = 4e67c866dd1424a9f00a24e70e6d86f2
```

Ako se promijeni samo jedno slovo u rečenici („a“ u „b“), sažetak je zbog efekta lavine potpuno drugačiji:

```
MD5("Ovo je primjer teksta za dobivanje hash-b.")
  = 6d1f8ad5424ea77d9f7c74c1bdd4d2b1
```

Sažetak praznog niza znakova je sljedeći:

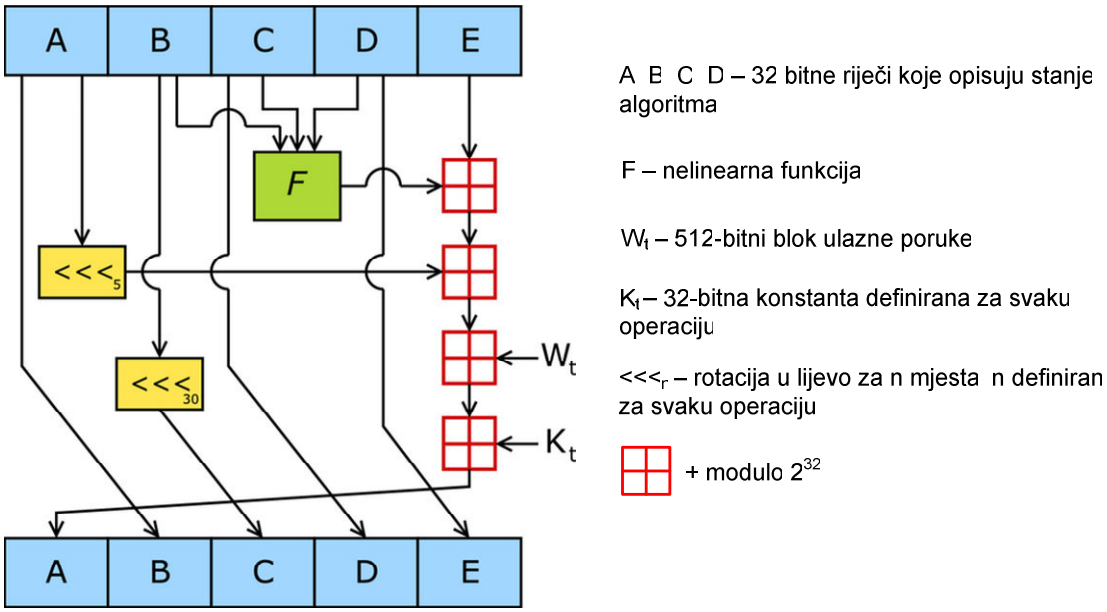
```
MD5("")
  = d41d8cd98f00b204e9800998ecf8427e
```

MD5 se često koristi za provjeru integriteta datoteka pomoću sažetka datoteke – prilikom primitka datoteke izračunava se njen MD5 sažetak i uspoređuje se sa sažetkom koji je dobiven uz datoteku pa ako su identični, onda je datoteka nepromijenjena.

Druga najčešća primjena MD5 algoritma je za enkripciju kod pohranjivanja zaporki, no postoji velik broj MD5 reverznih baza podataka kojima se tako zaštićene zaporke mogu dekriptirati. Kao zaštita od takvih napada preporučljivo je prije kriptiranja u tekst zaporke dodati poznati slučajni niz bitova što će znatno otežati napad. Isto tako preporučljivo je MD5 enkripciju u takvim situacijama koristiti u više iteracija što također povećava nivo sigurnosti.

4.2. SHA-1 algoritam

SHA-1 algoritam (eng. *Secure Hash Algorithm 1*) je drugi iz niza SHA algoritama razvijenih od strane američke vladine agencije NSA (*National Security Agency*). SHA-1 je najpopularniji od svih SHA algoritama i koristi se u velikom broju današnjih aplikacija i sigurnosnih protokola.
Prvi SHA algoritam objavljen je 1993. godine, ali je ubrzo nakon objavljivanja povučen iz upotrebe zbog otkrivene mane, pa je tako 1995. godine objavljen SHA-1 algoritma koji u biti predstavlja samo malo izmijenjenu varijantu SHA-0 algoritma. SHA-1 algoritam smatra se nasljednikom MD5 algoritma i po mnogočemu mu je sličan. Princip rada SHA-1 algoritma prikazan je na sljedećoj slici.



Slika 8: Princip rada SHA-1 algoritma

SHA-1 algoritam uzima kao ulaz poruku maksimalne dužine 2^{64} bita i iz nje izračunava sažetak dužine 160 bita. Princip rada također se bazira na iteracijama (kroz koje se mijenja nelinearna funkcija F) koje mijenjaju stanje algoritma.

Primjer SHA-1 sažetka:

```
SHA1("Ovo je primjer teksta za dobivanje hash-a.")
  = 32269d26f344355d3d35cae1265607048e54a1e7
```

Ako se promijeni samo jedno slovo u rečenici („a“ u „b“), sažetak je zbog efekta lavine potpuno drugačiji:

```
SHA1("Ovo je primjer teksta za dobivanje hash-b.")
  = 4314c5a6249243712014f094c7b4a73eb6eb2be6
```

Sažetak praznog niza znakova je sljedeći:

```
SHA1("")
  = da39a3ee5e6b4b0d3255bfef95601890afd80709
```

SHA-1 algoritam je također dokazano probijen, ali je za ostvarenje proboja potrebno korištenje kompleksnih analitičkih funkcija za koje se sumnja da su dostupne Internet korisnicima. Unatoč toj relativnoj sigurnosti, upotreba SHA-1 algoritma u budućim aplikacijama je upitna. Kao poboljšani nasljednik SHA-1 algoritma predstavljen je SHA-2 algoritam koji sadrži neke izmjene u odnosu na SHA-1 i kao rezultat daje duži sažetak i to u nekoliko varijanti SHA-224, SHA-256, SHA-384 i SHA-512, gdje brojke u nazivu označavaju dužinu sažetka. Ovi algoritmi pošto nisu u široj upotrebi nisu bili predmet detaljnijih ispitivanja, ali zasad za njih ne postoje dokazani proboji.

4.3. RIPEMD algoritam

RIPEMD (eng. *RACE Integrity Primitives Evaluation Message Digest*) algoritam je kriptografski algoritam za izračunavanje sažetka objavljen 1996. godine od strane tri autora iz Evrope: Hans Dobbertin, Antoon Bosselaers i Bart Preneel. RIPEMD algoritam bazira se na istim principima kao i MD4 algoritam, a po sigurnosnim svojstvima je sličan popularnijem SHA-1 algoritmu.
Postoje 4 varijante RIPEMD algoritma: RIPEMD-128, RIPEMD-160, RIPEMD-256 i RIPEMD-320 pri čemu brojke u nazivu odgovaraju dužinama sažetka koje se algoritmima dobivaju. RIPEMD-128 algoritam je upitne sigurnosti dok algoritmi s dužim sažetkom nemaju povećan nivo sigurnosti već samo smanjenu mogućnost kolizije.
Osnovna razlika između SHA-1 i RIPEMD algoritma je u tome što je RIPEMD algoritam razvijen od strane akademske zajednice pa prema tome njegova upotreba nije ograničena patentima. RIPEMD zbog svoje manje popularnosti nije toliko proučavan kao SHA-1 algoritam, ali je i za njega, tj. za originalni RIPEMD dizajn, pronađen proboj 2004. godine.

4.4. MAC algoritam

MAC (eng. *Message Authentication Code*) je algoritam koji se upotrebljava za autentikaciju i provjeru integriteta poruka tako što se uz poruku šalje i dodatni podatak koji se naziva MAC sažetak i koji se dobiva upotrebom MAC algoritma i tajnog ključa. Primatelj poruke može uz posjed identičnog tajnog ključa upotrebom istog algoritma provjeriti ukoliko MAC sažetak odgovara primljenoj poruci te time može verificirati integritet i autentičnost poruke.
Treba napomenuti kako MAC algoritam nije identičan digitalnom potpisu jer se za verifikaciju koristi isti ključ kojim se i izračunava sažetak. Zbog istog razloga MAC sažetak se ne može koristiti kao dokaz neporecivosti poruke jer svaki primatelj poruke posjeduje tajni ključ kojim može generirati novu poruku i novi MAC sažetak.
MAC algoritmi imaju striktne sigurnosne zahtjeve koji specificiraju da takav algoritam mora biti siguran od krivotvorenja, tj. da nitko ne smije biti u stanju generirati važeći MAC sažetak za neku poruku M bez obzira na sadržaj poruke. To znači da ako netko uspije generirati važeći MAC sažetak za bilo kakvu poruku (pa makar njen sadržaj nema nikakvog smisla) korištenjem nekog ključa, onda algoritam nije siguran.
Da bi se povećala sigurnost MAC algoritama oni se nadograđuju upotrebom drugih kriptografskih algoritama kao što su na primjer MD5 ili SHA-1 – takvi algoritmi se onda nazivaju prema funkciji koja se koristi, npr. HMAC-MD5 ili HMAC-SHA-1.

4.5. Ostali oblici implementacija

Osim opisanih postoje i druge manje poznate implementacije algoritama za izračunavanje sažetka:

- Snefru algoritam – autor Ralph Merkle; generira 128 ili 256 bitni sažetak,
- Tiger algoritam – autori Ross Anderson i Eli Biham; predviđen za 64-bitne procesore; generira 128-bitni, 160-bitni ili 192-bitni sažetak,
- Whirlpool – autori Vincent Rijmen i Paulo S. L. M. Barreto (postoje 3 varijante Whirlpool-0, Whirlpool-T, Whirlpool); baziran na Merkle-Damgard konstrukciji i Miyaguchi-Preneel kompresijskoj funkciji; generira 512-bitni sažetak; prihvaćen kao ISO/IEC 10118-3:2004 standard,
- HAVAL – autori Yuliang Zheng, Josef Pieprzyk, i Jennifer Seberry; generira 128-bitni, 160-bitni, 192-bitni, 224-bitni i 256-bitni sažetak.

5. Ostale primjene algoritma za izračunavanje sažetka

5.1. *Hash* tablice

Hash tablice predstavljaju jednu od glavnih primjena algoritama za izračunavanje sažetka izvan kriptografije. *Hash* tablice omogućavaju brzo pronalaženje podataka prema njegovom ključu koji predstavlja njegov sažetak.

Kod ovakve primjene najvažnije svojstvo algoritma je što postoji vrlo mali broj ili uopće ne postoje kolizije, tj. svaki podatak u tablici ima jedinstveni ključ/sažetak. Time se omogućava brzi dohvat podatka iz prvog ili iz vrlo malog broja pokušaja.

Algoritmi koji se upotrebljavaju u kriptografiji i čija je glavna karakteristika što se ponašaju kao funkcija generiranja slučajnih brojeva, obično imaju mali broj kolizija, ali je njihovo potpuno izbjegavanje gotovo nemoguće. No zato je za napadače koji žele naći kolizije to gotovo nemoguće učiniti.

S druge strane heurističkim algoritmima za izračunavanje sažetka može se postići manji broj kolizija nego kriptografskim algoritmima i to iskorištavanjem regularnosti kod predvidivih ulaznih podataka. Npr. za potrebe *hash* tablice može se razviti heuristički algoritam koji će za dana imena datoteka FILE0000.CHK, FILE0001.CHK, FILE0002.CHK generirati sažetke koji će slijediti jedan drugoga i time neće doći do kolizija. Međutim takvi algoritmi, iako se odlično ponašaju za slijedne podatke, mogu se ponašati vrlo loše za slučajne vrijednosti ulaznih podataka i dovesti do velikog broja kolizija na što treba paziti prilikom dizajna algoritma.

Druga bitna stvar na koju je potrebno paziti kod dizajna algoritma za *hash* tablice je brzina algoritma. Pošto se *hash* tablice uglavnom koriste za minimiziranje vremena potrebnog za pronalaženje nekog podatka prema njegovom sažetku, onda je za dohvaćanje podatka, tj. za izračun sažetka, važno da bude izrazito brz.

Jedan od najpoznatijih algoritama korištenih u *hash* tablicama je LOOKUP2 algoritam koji radi odlično, ali je zato vrlo jednostavan i lako reverzibilan te je kao kriptografski algoritam neupotrebljiv.

5.2. Detekcija pogrešaka

Algoritmi za izračunavanje sažetka mogu se koristiti za detekciju i korekciju grešaka nastalih u prijenosu ili pohranjivanju podataka. Detekcija grešaka obično se obavlja pomoću dodatnih bitova koji se šalju ili pohranjuju uz podatke koji se štite od grešaka. Takvi dodatni bitovi, koji u stvari predstavljaju neku vrstu sažetka podatka, opisuju neki parametar podatka koji se nakon prijenosa ili pohrane provjerava i uspoređuje sa stvarnim stanjem podatka – ako stanje odgovara onom koje opisuju dodatni bitovi, tada je podatak neoštećen.

Metode za detekciju grešaka i algoritmi za izračun takvih sažetaka mogu biti vrlo jednostavne kao što su na primjer sljedeće:

- slanje dodatnih bitova koji opisuju dužinu podatka – ne detektira greške ako je podatak ostao iste dužine,
- slanje dodatnih bitova koji opisuju paritet podatka (broj binarnih jedinica) – ne detektira pogrešku ako se promijenio paran broj bitova.

Postoje i kompleksnije i efektivnije metode za detekciju grešaka koje uzimaju u obzir ne samo vrijednost pojedinih bitova podatka već i njihovu poziciju, a jedna od najpopularnijih je CRC (eng. *Cyclic Redundancy Check*) algoritam.

5.2.1. CRC algoritam

CRC (eng. *Cyclic Redundancy Check*) algoritam bazira se na postupku dijeljenja polinoma i pamćenja njihovih koeficijenata. Svaki podatak može se zapisati kao skup koeficijenata polinoma određenog reda. Ako se taj polinom podijeli s drugim polinomom čija vrijednost je fiksna, a prije dijeljenja pomnoži s X^n gdje n određuje red fiksnog polinoma, onda koeficijenti polinoma dobivenog dijeljenjem predstavljaju CRC sažetak podatka.

Izračun se može pokazati na primjeru:

- $x^2 + x + 1$ – originalni podatak = 111
- $x + 1$ – fiksni polinom, tj. ključ reda 1
- $x^3 + x^2 + x$ – podatak pomnožen s x^1 , gdje je 1 red ključa
- $CRC = 1 = x^0$ = ostatak nakon dijeljenja $(x^3 + x^2 + x) / (x + 1)$

Općenita formula za izračun:

$$M(x) * x^n = Q(x) * K(x) + R(x)$$

gdje je:

- M(x) – polinom originalnog podatka,
- K(x) – polinom ključa reda n,
- M(x) X^n – originalna poruka s dodanih *n* nula na kraju,
- R(x) – CRC sažetak.

CRC algoritmi su vrlo popularni zbog toga što ih je izrazito jednostavno implementirati u binarnom obliku i jednostavni su za matematičku analizu, a istovremeno su jako efikasni u detekciji čestih pogrešaka uzrokovanih šumom u komunikacijskom kanalu.

5.3. Audio identifikacija

Algoritmi za izračunavanje sažetka mogu se upotrebljavati i nad audio datotekama, npr. mp3 formata. Za to se mogu upotrebljavati konvencionalni algoritmi poput MD5 algoritma, ali bi takvi algoritmi vrlo vjerojatno bili osjetljivi na uobičajene transformacije koje se javljaju u audio sustavim kao što su vremenski pomaci, greške u zapisu na CD-u, promjena glasnoće, i sl. Za usporedbu audio datoteka razvijeni su posebni algoritmi kojima se mogu naći sve datoteke koje su identične iz perspektive ljudskog uha pa tako postoje algoritmi i usluge koje se na njima baziraju kojima se na osnovu reprodukcije zvuka iz zvučnika mogu dobiti podaci kojoj se pjesmi radi.

6. Zaključak

Kao što je vidljivo iz dokumenta, algoritmi za izračunavanje sažetka imaju široku primjenu najvećim dijelom u kriptografiji, ali i u drugim područjima. Razvoj takvih algoritama je gotovo neprestan jer se za svaki od algoritama može sa sigurnošću utvrditi da će nakon nekog vremena koje provede u aktivnoj upotrebi, biti provaljen zbog stalnog napretka tehnologije i mogućnosti matematičkih analiza koje je moguće napraviti u jedinici vremena.
Ali s druge strane isti napredak koriste i dizajneri algoritama za izračunavanje sažetka koji kreiraju sve kompleksnije algoritme koji zbog sve veće brzine računala postaju dovoljno brzi kako bi bili primijenjeni u svakodnevnoj komunikaciji pa izgleda da pobjednika u toj trci nema. Jedini gubitnik u trci je potrošač koji kako bi osigurao traženi nivo sigurnosti mora stalno pratiti i razvoj algoritama i prateće tehnologije i za to uglavnom mora izdvojiti značajna sredstva.

7. Reference

[1] Hash Functions and Block Ciphers, http://www.burtleburtle.net/bob/hash/, kolovoz 2006.
[2] RSA security labs, http://www.rsasecurity.com, kolovoz 2006.
[3] Whirlpool, http://paginas.terra.com.br/informatica/paulobarreto/WhirlpoolPage.html, kolovoz 2006.
[4] Bruce Schneier, Applied Cryptography: Protocols, Algorithms, and Source Code in C, Second Edition, 1996.
[5] A. Menezes, P. van Oorschot, S. Vanstone: Handbook of Applied Cryptography, CRC Press, 1996.
[6] RFC 1321, The MD5 Message-Digest Algorithm, http://www.ietf.org/rfc/rfc1321.txt, kolovoz 2006.
[7] Primjeri implementacije MD5 i SHA-1, http://www.cs.eku.edu, kolovoz 2006.