SHA algoritmi i njihova primjena

Dubravac, Marko

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA, OSIJEK

Sveučilišni studij

SHA ALGORITMI I NJIHOVA PRIMJENA

Završni rad

Marko Dubravac

Osijek, 2022.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 13.09.2022.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime Pristupnika:	Marko Dubravac
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. Pristupnika, godina upisa:	R 4338, 22.07.2019.
OIB Pristupnika:	43543801462
Mentor:	Izv. prof. dr. sc. Krešimir Grgić
Sumentor:	Ana Pejković, mag. ing. el.
Sumentor iz tvrtke:	
Naslov završnog rada:	SHA algoritmi i njihova primjena
Znanstvena grana rada:	**Telekomunikacije i informatika (zn. polje elektrotehnika)**
Zadatak završnog rad:	SHA skupinu hash algoritama (Secure Hash Algorithm) sačinjavaju danas najrašireniji hash algoritmi ugrađeni u brojne sigurnosne protokole i aplikacije. Potrebno je analizirati i objasniti ove algoritme, te područja i mogućnosti njihove primjene. Analizu provesti na primjeru implementacija različitih algoritama iz SHA skupine. Sumentor s FERIT-a: Ana Pejković
Prijedlog ocjene završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 2 razina
Datum prijedloga ocjene od strane mentora:	13.09.2022.
Datum potvrde ocjene od strane Odbora:	21.09.2022.

Potvrda mentora o predaji konačne verzije rada:	*Mentor elektronički potpisao predaju konačne verzije.*
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA O ORIGINALNOSTI RADA

Osijek, 21.09.2022.

Ime i prezime studenta:	Marko Dubravac
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R 4338, 22.07.2019.
Turnitin podudaranje [%]:	4

Ovom izjavom izjavljujem da je rad pod nazivom**: SHA algoritmi i njihova primjena**

izrađen pod vodstvom mentora Izv. prof. dr. sc. Krešimir Grgić

i sumentora Ana Pejković, mag. ing. el.

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

Sadržaj

1. UVOD

Povećanjem općeg broja korisnika internetskih usluga dolazi do povećanja zlonamjernih korisnika te onih koji pokušavaju iskoristiti sigurnosne propuste. Sigurnost u internetskoj komunikaciji dobiva na važnosti iz dana u dan. Iz toga razloga postoje sigurnosni mehanizmi u komunikaciji. Jedan od problema koji dolazi s tim mehanizmima je optimizacija brzine i sigurnosti kod prijenosa podataka. Hash algoritmi služe kao baza brojnim sigurnosnim mehanizmima jer se koriste za provjeru očuvanju integriteta podataka. U internetskoj komunikaciji, očuvanje integriteta predstavlja jednu od ključnih zadaća sigurnosti. Riječ je o očuvanju cijelosti i nepromijenjenosti podatkovnog bloka odnosno poruke koja se prenosi. Sigurnosni hash algoritmi ili SHA (engl. *Secure Hash Algorithms*) jedni su od najkorištenijih hash algoritama. Razlog tomu je njihova brzina te napredne tehnike stvaranja kriptiranih hash poruka. Zbog svoje zamršenosti daleko su ispred konkurencije, pa se koriste u sustavima koji zahtijevaju potpuni integritet podataka, kao na primjer u transakcijama kriptovalutama. Po nekim definicijama, sigurnost je nedovršen proces te zbog toga postoje više različitih sigurnosnih hash algoritama koji su se razvijali s povećanjem potrebe za sigurnošću.

Rad započinje s teorijski uvod o kriptografskim hash funkcijama u kojem je opisan način rada, primjena i sigurnosne potrebe funkcija kao što je SHA. Slijedi opis kriptografskih algoritama SHA s postupcima izrade pojedinih algoritama, njihova usporedba i kratak opis protokola u kojima se koriste. Za kraj je opisana izrada SHA-256 u programskoj platformi MATLAB.

1.1. Zadatak završnog rada

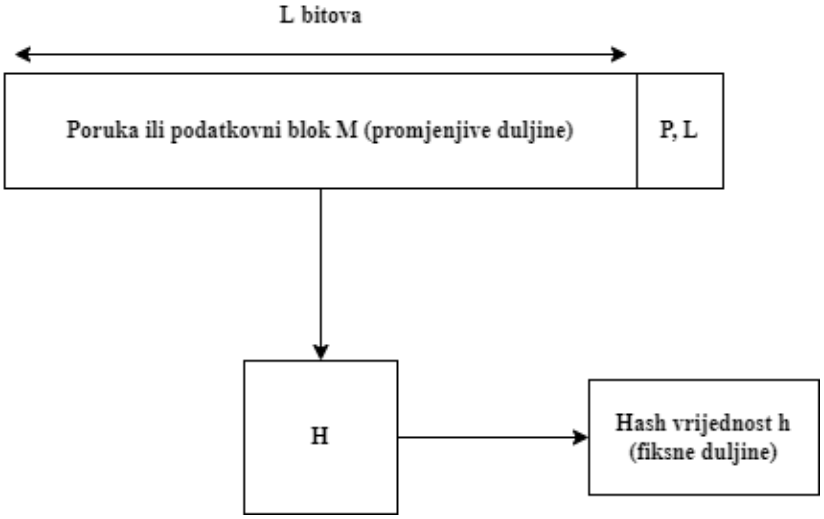
SHA skupinu hash algoritama (Secure Hash Algorithm) sačinjavaju danas najrašireniji hash algoritmi ugrađeni u brojne sigurnosne protokole i aplikacije. Potrebno je analizirati i objasniti ove algoritme, te područja i mogućnosti njihove primjene. Analizu provesti na primjeru implementacija različitih algoritama iz SHA skupine.

2. HASH FUNKCIJE

Prema [1], hash funkcija H, za ulazni podatkovni blok promjenjive duljine M na izlazu vraća hash vrijednost stalne duljine h=H(M), drugim riječima hash funkcije preslikavaju niz ulaznih podataka neodređene duljine u niz podataka čija je duljina fiksno određena. Podatkovni blok M još se naziva i preslika hash vrijednosti h. Autor [1] također navodi da je svojstvo „dobre“ hash funkcije da za veliki skup ulaza vraća izlaze koji su ravnomjerno raspoređeni i naizgled nasumični. Među raznim hash funkcijama, one koje zadovoljavaju određena svojstva nazivaju se kriptografske hash funkcije koje su pogodne za korištenje u kriptografiji. One su jednosmjerne (engl. *one–way*), odnosno hash vrijednost nije namijenjena za vraćanje u originalni podatkovni blok, a jedini način za određivanje originalne poruke je ručno pogađanje preslike koje se naziva napadom grubom silom (engl. *brute force attack*). Osnovna svrha primjene jednosmjerne hash funkcija je očuvanje integriteta podataka i prema [6] naširoko se koristi za digitalne potpise, provjeru autentičnosti poruke (MAC), metodu razmjene ključeva i slično. U ovom kontekstu očuvanje integriteta podataka znači da nije došlo do promjene podatkovnog bloka, kao na primjer manipuliranja porukom od strane osobe koja ne sudjeluje u komunikaciji. Promjena jednog bita podatkovnog bloka izaziva veliku vjerojatnost promjene hash vrijednosti. Prema [5] karakteristike idealne kriptografske hash funkcije su:

1. determinizam,
2. brzina,
3. jednosmjernost,
4. jedinstvenost po hash vrijednosti,
5. osjetljivost na promjene.

Na slici 2.1. je grafički prikaz kriptografske hash funkcije h=H(M). Ulazni blok u shemi je poruka ili podatkovni blok. L je duljina poruke u bitovima. Često se duljina ulazne poruke dopunjava na unaprijed određen broj, dok stvarna duljina može varirati te je zbog toga potrebna dopuna P (engl. *padding*). Dopunjena duljina ulazne poruke je unaprijed određen broj kako bi se otežalo potencijalnim napadačima u pokušaju stvaranja jednake hash vrijednosti [1], odnosno u stvaranju jednake ulazne vrijednosti.



Slika 2.1. *Kriptografska hash funkcija, h=H(M).* [1]

2.1. Jednostavan primjer kriptografske hash funkcije

Kriptografske hash funkcije mogu biti jednostavne, ali moraju pratiti opća načela izrade kriptografskih hash funkcija. Ulaz u obliku poruke ili datoteke, promatra se kao slijed n-bitnih blokova te se obrađuje blok po blok na iterativan način. Rezultat tog obrađivanja je n-bitna funkcija. Isključivo ILI logička operacija (XOR) svakog bita pojedinih blokova je jedan od najjednostavnijih primjera kriptografskih hash funkcija (2-1)[1] i zapisuje se kao:

$$C_i = b_{i1} \oplus b_{i2} \oplus \ldots \oplus b_{im} \qquad (2\text{-}1)$$

gdje je:

- C_i – i-ti bit hash vrijednosti (i je od 1 do n, odnosno od prvog do zadnjeg bita hash vrijednosti)
- m – broj n-bitnih blokova u ulaznoj vrijednosti
- b_{ij} – i-ti bit u j-tom bloku
- $\oplus$ – isključivo ILI operacija (XOR)

Ova funkcija je učinkovita za provjeru integriteta slučajnih podataka i proizvodi jednostavan paritet za svaki položaj bita i naziva se longitudinalna provjera redundancije [1]. S ovom hash funkcijom svaka n-bitna hash vrijednost jednako je moguća, zbog toga vjerojatnost da se hash vrijednost ne promjeni prilikom podatkovne pogreške iznosi 2^{-n}. U slučaju očekivanja određenih podataka funkcija je smanjene učinkovitosti. Dodavanje jednobitnog kružnog pomaka na svaki

blok kod stvaranja hash vrijednosti je jednostavan način za poboljšanje učinkovitosti. Taj postupak se naziva rotacija i on se obavlja se u sljedećim koracima [1]:

1. Početna n-bitna hash vrijednost se postavlja na nulu
2. Svaki sljedeći n-bitni blok podataka pomiče trenutnu hash vrijednost u lijevo te se izvodi isključivo ili operacija koja podatkovni blok pretvara u hash vrijednost

Rotacija stvara dojam nasumičnosti u hash vrijednosti, što dovodi do dobrog integriteta podataka, ali ne čini ništa po pitanju sigurnosti poruka kada je riječ o nešifriranim informacijama. Jednostavnom pripremom očekivane poruke, podjelom na n-bitne blokove te primjenom rotacijske XOR (RXOR) operacije može se dobiti željena hash vrijednost. Kriptografske hash funkcije s XOR ili RXOR operacijama, iako jednostavne, mogu se koristiti u slučajevima gdje su ulazna poruka i hash vrijednost šifrirani.

2.2. Primjena kriptografskih hash funkcija

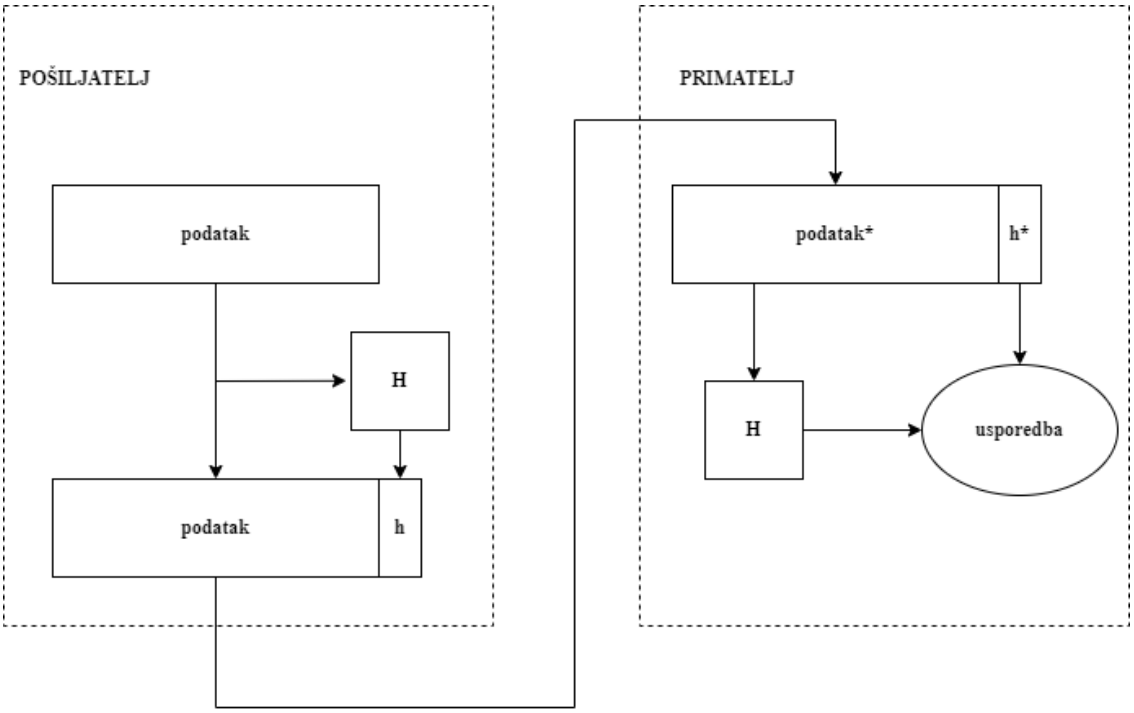
Kriptografske hash funkcije imaju široko područje primjene te zbog toga se koriste kod brojnih sigurnosnih aplikacija i internetskih protokola [1]. U nastavku je opisana njihova primjena kod sustava za provjeru autentičnosti poruka i kod digitalnih potpisa.

2.2.1. Autentičnost poruka

Za provjeru integriteta poruka koriste se mehanizmi ili usluge za provjeru autentičnosti poruka. Brojni sustavi zahtijevaju mehanizam za provjeru autentičnosti, koji osigurava valjani identitet pošiljatelja prilikom komunikacije. Hash vrijednost funkcije koja se koristi za pružanje autentičnosti poruke naziva se sažetak poruke (engl. *message digest*). Provjera autentičnosti poruke korištenjem hash funkcije izvodi se na način da pošiljatelj izračuna hash vrijednost na temelju bitova u poruci. Nakon toga, poruka i hash vrijednost se prenose. Primatelj izvodi isti hash izračun na bitovima dobivene poruke te uspoređuje dvije hash vrijednosti, onu koju je izračunao pošiljat s onom koju je on izračunao. U slučaju nepodudaranja došlo je do izmjene u bitovima poruke ili eventualno hash vrijednosti.

Slika 2.2. grafički prikazuje sustav za provjeru autentičnosti poruke, gdje je H hash funkcija, podatak poruka koju šalje pošiljatelj, h hash vrijednost koju je izračunao pošiljatelj, podatak*

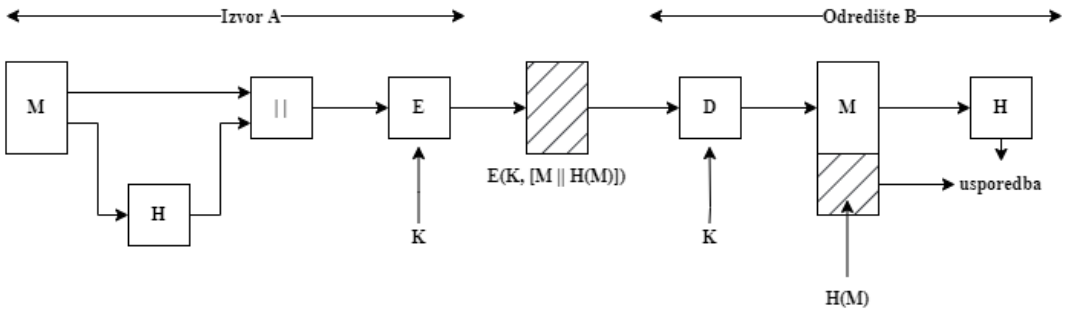
poruka koju je primio primatelj te h* hash vrijednost koju je izračunao primatelj. Sustav se još sastoji od mehanizma za usporedbu hash vrijednosti.



***Slika 2.2**. Sustav za provjeru autentičnosti poruke.*

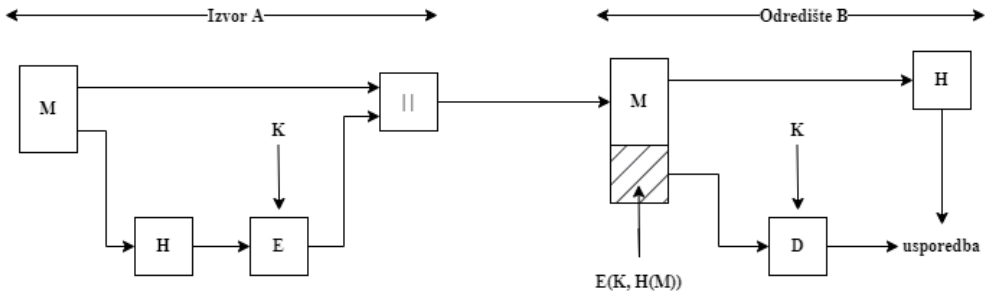
Hash funkcija trebala bi se poslati na siguran način. Ako je hash funkcija zaštićena, u slučaju da osoba koja ne sudjeluje u komunikaciji presretne poruku te ju izmjeni, ne bi mogla proizvesti valjanu hash vrijednost. Kod usporedbe bi se pojavilo nepodudaranje. Neki od načina očuvanja autentičnosti sigurnim slanjem su [1]:

a) Šifriranje poruke zajedno s pripadajućom hash vrijednosti. Slika 2.3. predstavlja sustav u kojem se poruka M i pripadajuća hash vrijednost H šifriraju simetričnom enkripcijom E. Simetrična enkripcija je način zaštite podataka pomoću tajnog ključa, odnosno lozinke za pristup informacijama. Na slici pošiljatelj (izvor A) i primatelj (odredište B) dijele tajni ključ K. Primatelj dešifrira poruku i hash vrijednost, izračunava vlastitu hash vrijednost te uspoređuje te dvije vrijednosti. Ovaj sustav pruža povjerljivost jer je poruka šifrirana.



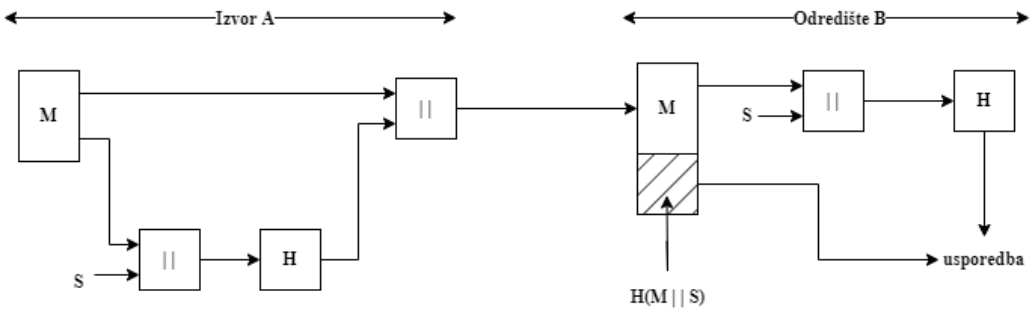
Slika 2.3. *Sustav s enkripcijom poruke i hash vrijednosti.* [1]

b) Šifriranje hash vrijednosti prije slanja s pripadajućom porukom. Na slici 2.4. je sustav u kojem je samo hash vrijednost šifrirana simetričnom enkripcijom. Poruka M i šifrirana hash vrijednost E(K,H(M)), gdje je K ključ, dolaze do primatelja. Šifrirana hash vrijednost se dešifrira te uspoređuje s izračunatom hash vrijednosti. Ovim postupkom se smanjuje vrijeme obrade, ali se ne pruža povjerljivost poruke.



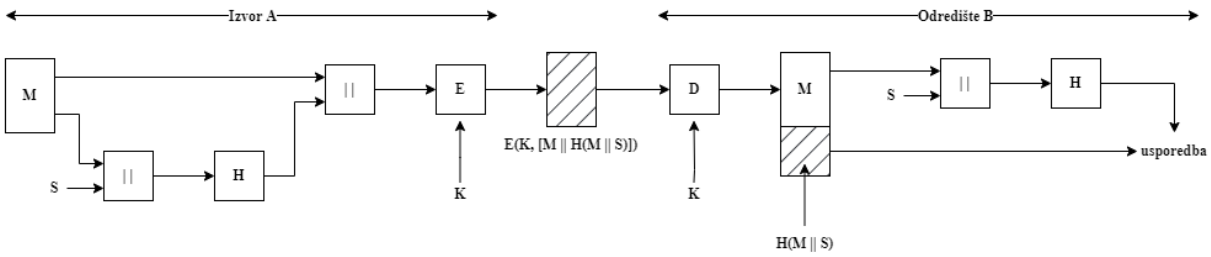
Slika 2.4. *Sustav s enkripcijom hash vrijednosti.* [1]

c) Korištenje tajne vrijednosti bez enkripcije. Slika 2.5. predstavlja sustav s tajnom vrijednosti S, koja je poznata pošiljatelju i primatelju. Na izvoru A poruka M i tajna vrijednost S se ulančavaju te generiraju hash vrijednost H pomoću hash funkcije. Na odredištu poruka se ulančava s tajnom vrijednosti, izračunava se hash vrijednost te uspoređuju se s dobivenom hash vrijednosti. Poruke su sigurne od modificiranja od treće strane jer se tajna vrijednost ne šalje.



Slika 2.5. *Sustav s tajnom vrijednosti bez enkripcije.* [1]

d) Korištenje tajne vrijednosti s enkripcijom. Dodavanjem enkripcije na čitavu poruku i hash vrijednost metodi c) može se pružiti povjerljivost poruke M, kao što je vidljivo iz sustava na slici 2.6.



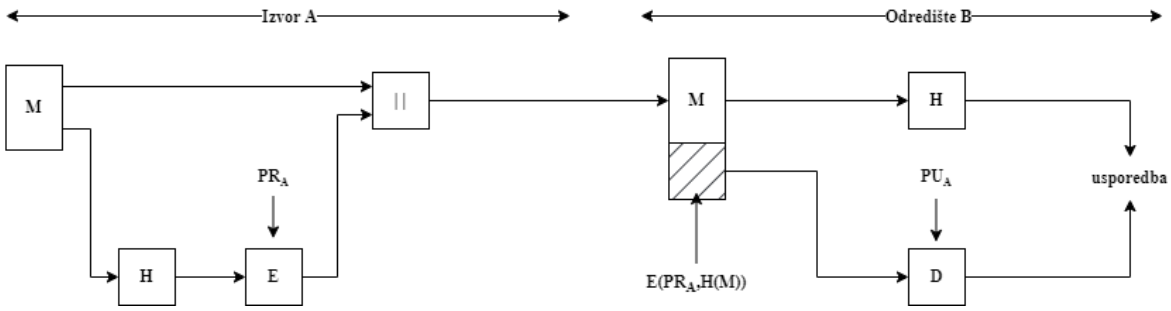
Slika 2.6. *Sustav s tajnom vrijednosti s enkripcijom.* [1]

Metoda b) se koristi kada nije potrebna povjerljivost poruke. Metode a) i d), iako pružaju povjerljivost, usporavaju sustav enkripcijom čitave poruke. Postoji više razloga za izbjegavanje enkripcije [7]. Softveri za enkripciju su relativno spori jer se u komunikaciji radi o stalnom protoku podataka. Troškovi dodatnog sklopovlja za enkripciju nisu zanemarivi. Sklopovlje je optimizirano za velike podatkovne blokove te pri slanju manjih nezanemariv dio vremena se potroši na povezivanje.

Za provjeru autentičnosti često se koristi kod provjere autentičnosti (engl. *message authentication code*), odnosno MAC. Kod tog načina, strane u komunikaciji dijele tajni ključ. Pošiljatelj koristi MAC funkciju, koja s podatkovnim blokom i tajnim ključem, stvara hash vrijednost, odnosno MAC vrijednost. Primjenom MAC funkcije na cijelu poruku, primatelj može stvoriti vlastitu MAC vrijednost te ju usporediti s dobivenom. Time dolazi do očuvanja integriteta poruke jer je za promjenu potrebno poznavati tajni ključ. U praksi su MAC algoritmi učinkovitiji od enkripcijskih.

2.2.2. Digitalni potpisi

Digitalni potpisi rade na sličan način kao i MAC. Razlika je u tome što kod digitalnih potpisa postoje dva ključa, javni i privatni. Hash vrijednost poruke je šifrirana privatnim ključem pošiljatelja, a svatko tko zna javni ključ može provjeriti integritet poruke. Privatni ključ poznat je samo pošiljatelju. Osobi koja želi promijeniti originalnu poruku gotovo je nemoguće proizvesti šifriranu hash vrijednost, koja će kod dešifriranja s javnim ključem pošiljatelja proizvesti jednaku hash vrijednost koju je izračunao primatelj. Slika 2.7. predstavlja princip rada digitalnih potpisa. Sličan je kao sustav za provjeru autentičnosti s šifriranjem hash vrijednosti sa slike 2.4., gdje je jedan ključ za šifriranje i dešifriranje K zamijenjen s privatnim ključem pošiljatelja PR_A za šifriranje i javnim ključem pošiljatelja PU_A za dešifriranje.



Slika 2.7. *Princip rada digitalnih potpisa.* [1]

2.2.3. Druge primjene

Hash funkcije se koriste kod jednosmjernih datoteka s lozinkama. Zlonamjerna osoba koja bi pristupila datoteci s lozinkama u tom slučaju ne može vidjeti lozinke, nego samo njihove hash vrijednosti. Većina operativnih sustava koristi ovaj pristup zaštite lozinki [1].

Još jedna primjena hash funkcija je zaštita od upada i virusa. Za sve datoteke na sustavu stvaraju se hash vrijednosti te se spremaju na siguran vanjski izvor. Hash vrijednosti datoteka sustava i vanjskog izvora se mogu usporediti te kod nepodudaranja može se utvrditi da je došlo do izmjene u datotekama sustava [1].

Za konstruiranje pseudoslučajne funkcije ili generatora pseudoslučajnih brojeva također se mogu koristiti kriptografske hash funkcije [1].

2.3. Sigurnosne potrebe kriptografskih hash funkcija

Budući da je kod hash funkcija h=H(x) više naprema jedan preslikavanje, različiti ulazni podatkovni blokovi ili preslike mogu rezultirati jednakim hash vrijednostima. Kolizija nastaje kada za dvije različite preslike x i y, upotrebom iste hash funkcije, nastaju dvije jednake hash vrijednosti H(x)=H(y). Količina preslika za danu hash vrijednost služi kao mjera potencijalnih kolizija. Za n bitnu hash vrijednost i b bitnu ulaznu poruku, gdje je b > n, postoji 2^b mogućih poruka i 2^n mogućih hash vrijednosti. Prosječna količina preslika za svaku hash vrijednost iznosi 2^{b-n}. Kod ulaznih poruka proizvoljne duljine broj preslika ovisi o toj duljini. Sigurnosne potrebe kriptografskih funkcija su: otpornost na preslike ili jednosmjernost, otpornost na druge preslike ili slaba otpornost na kolizije, otpornost na kolizije ili jaka otpornost na kolizije i pseudoslučajnost. Hash funkcije ne moraju zadovoljavati sve sigurnosne potrebe. One koje zadovoljavaju svojstvo jake otpornosti na kolizije nazivaju se još i jake hash funkcije.

2.3.1. Jednosmjernost

Otpornost na preslike ili jednosmjernost je sigurnosna potreba kriptografskih hash funkcija koja nalaže da je za bilo koju hash vrijednost h nemoguće računskim putem pronaći njenu presliku y, tako da je H(y)=x. Jednosmjernost je posebno važna kod sustava za provjeru autentičnosti s tajnom vrijednosti. Ona se može lako otkriti ako se iz hash vrijednosti može dobiti poruka. Kod presretanja prijenosa se može dobiti poruku M i hash vrijednost h=H(M||S). Upotrebom inverza H^{-1}(M||S) može se dobiti M||S te poznavanjem poruke M lako je pronaći tajnu vrijednost S.

2.3.2. Slaba otpornost na kolizije

Svojstvo otpornosti na druge preslike ili slaba otpornost na kolizije je sigurnosno svojstvo koje osigurava da je nemoguće pronaći alternativnu poruku s jednakom hash vrijednosti kao s danom porukom. Drugim riječima, za bilo koji podatkovni blok x, matematički je nemoguće pronaći y≠x, takav da vrijedi H(x)=H(y) [1]. Ovo svojstvo sprječava krivotvorenje kod sustava s šifriranom hash vrijednosti. Kada se prijenos poruke s šifriranom hash vrijednosti presretne, za poruku se može generirati nešifrirana hash vrijednost, pronaći alternativna poruka s jednakom

hash vrijednosti te poslati alternativna poruka s originalnom šifriranom hash vrijednosti. Kod tog slučaja primatelj ne može znati da je došlo do promjene poruke.

2.3.3. Jaka otpornost na kolizije

Otpornost na kolizije ili jaka otpornost na kolizije je sigurnosno svojstvo koje čini hash funkcije jakima. Matematički ne smije postojati niti jedan par x i y za koji vrijedi da je H(x)=H(y) da bi svojstvo bilo ispunjeno. Jake hash funkcije štite kod napada gdje jedna strana generira poruku drugoj za potpis. Prije slanja mogu se stvoriti dvije potpuno različite poruke s jednakim hash vrijednostima. Osoba koja potpisuje može potpisati jednu stvar, dok osoba koja generira poruku može na temelju hash vrijednosti tvrditi da je potpisana sasvim druga stvar. To je svojstvo izrazito važno kod novčanih transakcija.

2.3.4. Pseudoslučajnost

Pseudoslučajnost nije nužna sigurnosna potreba ali se podrazumijeva kada se priča o kriptografskim hash funkcijama. Ključne zadaće kriptografskih hash funkcija kao što su stvaranje ključeva, generiranje pseudoslučajnih brojeva i pružanje integriteta podataka, sve zahtijevaju nasumičan izlaz te dojam slučajnosti. Iako u teoriji nije potrebna, kod primjene je često tražena osobina.

3. SIGURNOSNI HASH ALGORITMI (SHA)

Sigurnosni Hash Algoritmi (SHA) su jedni od najkorištenijih kriptografskih hash algoritama. Razlog tome je njihova učinkovitost [5]. Sigurni su jer je računalno nemoguće pronaći poruku koja odgovara danoj hash vrijednosti ili pronaći dvije poruke s jednakim hash vrijednostima, a svaka promjena poruke u prijenosu rezultirat će u promjeni danog potpisa. Do 2005. su u gotovo svakom drugom naširoko korištenom hash algoritmu pronađeni sigurnosni propusti te je dugi niz godina SHA više-manje bio jedini standardizirani hash algoritam.

Prva verzija SHA izdana je 1993. godine kao dio saveznih standarda obrade informacija FIPS (engl. *Federal Information Processing Standards*) od strane američkog instituta za standarde i tehnologiju NIST-a (engl. *National Institute of Standards and Technology*) pod nazivom Secure Hash Standard (SHS) ili FIPS PUB 180. Revidirana verzija izdana je 1995. godine pod nazivom SHA-1, dok je prva verzija kasnije nazvana SHA-0. Obje se temelje na MD4 hash funkciji, a jedina razlika između SHA-0 i SHA-1 je dodatna operacija rotacije u proširenju poruka kod SHA-1 koja pruža dodatnu sigurnost [8]. U originalnoj dokumentaciji, moguća primjena je navedena u elektroničkom prijenosu sredstava, distribuciji programske podrške, pohrani podataka i drugim sustavima koji zahtijevaju osiguranje integriteta podataka i provjeru autentičnosti podataka, kao i u sustavima u kojima je potrebno stvoriti sažetu verziju poruke [2], a u zadnje vrijeme se koriste u raznim kriptovalutama kao što je Bitcoin. Mogu se implementirati u programskoj podršci, sklopovlju i ugrađenim programima, kao i u njihovim kombinacijama. Zadnja verzija standarda izdana je 2015. godine pod nazivom FIPS PUB 180-4 [3]. SHA-2 je najraširenija podskupina SHA algoritama. Sastoji se od SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 i SHA-512/256 algoritama.

Svaki SHA sastoji se od predobrade (engl. *preprocessing*) i hash računanja. Predobrada obuhvaća dopunjavanje poruke, rastavljanje dopunjene poruke na m-bitne blokove i inicijalizaciju vrijednosti potrebnih za hash računanje. Poruka se dopunjava da bi se osigurala da je ona višekratnik broja 512 ili 1024, ovisno o algoritmu. Rastavljanje dopunjene poruke je u svrhu stvaranja N m-bitnih blokova, odnosno blokova nad kojima se vrši hash računanje. Inicijalizacijom vrijednosti se postavljaju početne hash vrijednosti. Kod hash računanja generira se raspored poruke iz dopunjene poruke te se zajedno s njim, konstantama, operacijama na riječima i funkcijama iterativno generira niz hash vrijednosti. Konačna generirana hash vrijednost koristi se za određivanje sažetka poruke.

Slika 3.1. prikazuje izlaze, odnosno hash vrijednosti za testne ulazne poruke nakon obrade SHA-1 algoritmom. Iz slike se može zaključiti kako se hash vrijednosti dramatično razlikuju za male izmjene ulazne poruke te kako su one jednake duljine neovisno o duljini ulazne poruke.

Slika 3.1. *Testni izlazi nakon SHA-1 funkcije.* [5]

3.1. Predobrada

Predobrada kod SHA-1 i SHA-2 sastoji se od tri dijela: dopunjavanja poruke, rastavljanja dopunjene poruke na m-bitne blokove i postavljanja početnih hash vrijednosti. Postoje dva tipa SHA koji se razlikuju u dopunjavanju poruke i rastavljanju na m-bitne blokove:

a) Algoritmi s duljinom dopunjene poruke koja je višekratnih od 512 bita – SHA-1, SHA-224, SHA-256

b) Algoritmi s duljinom dopunjene poruke koja je višekratnik od 1024 bita – SHA-512, SHA-384, SHA-512/224 i SHA-512/256

U slučaju a) dopuna se radi na sljedeći način: na kraj poruke M dodaje se bit „1“, nakon kojeg slijedi k „0“ bitova, gdje je k najmanje ne-negativno rješenje jednadžbe (3-1) koja glasi:

$$L + 1 + k = 448 \; mod \; 512 \tag{3-1}$$

gdje je:

- L – duljina poruke M u bitovima
- *mod* – modularna operacija, odnosno operacija koja vraća ostatak cjelobrojnog dijeljenja

Nakon niza „0“ bitova dodaje se 64-bitni blok koji je jednak binarnom zapisu broja L. Na primjer, riječ TEST se sastoji od 32 bita u binarnom zapisu te jednadžbom (3-1) može se izračunati k kao k=448-(32+1)=415. Dopunjena poruka u ovom primjeru bi se sastojala od 32-bitne poruke, jednog bita „1“, 415 bitova „0“, te 64-bitnog zapisa broja 32 koji predstavlja duljinu početne poruke L. Konačna dopunjena poruka sastoji se od 512 bita.

Slučaj b) razlikuje se od slučaja a) u jednadžbi (3-2) što mijenja broj bitova konačne dopune, a ona glasi:

$$L + 1 + k = 896 \; mod \; 1024 \tag{3-2}$$

varijable i operacije su jednake kao kod jednadžbe (3-1). Nakon toga se dodaje 128-bitni blok koji je jednak binarnom zapisu broja L. Na primjer, za 32-bitnu riječ TEST može se izračunati vrijednost k kao k=896-(32+1)=863. Dopunjena poruka ovog primjera sastojala bi se od 32-bitne riječi, jednog bita „1“, 863 bita „0“, te 128-bitnog zapisa broja 32. Dopunjena poruka je tada 1024-bitna.

Kod a) dopunjene poruke se rastavljaju na N 512-bitnih blokova $M^{(1)}$, $M^{(2)}$,…,$M^{(N)}$. Ti se blokovi dijele još i na šesnaest 32-bitnih riječi, od kojih se prvih 32 bita bloka i označava s $M_0^{(i)}$, zatim $M_1^{(i)}$, pa sve do $M_{15}^{(i)}$.

Kod b) poruke se rastavljaju na N 1024-bitnih blokova $M^{(1)}$, $M^{(2)}$,…,$M^{(N)}$, a oni se još dijele na šesnaest 64-bitnih riječi.

3.2. SHA-1

SHA-1 se koristi za stvaranje hash vrijednosti na poruci M, koja je duljine L bita, gdje je $L < 2^{64}$. Koristi rasporeda osamdeset 32-bitnih riječi, pet radnih varijabli od 32 bita i hash vrijednost od pet 32-bitnih riječi. Za rezultat vraća 160-bitni sažetak poruke. Riječi rasporeda označavaju se s W_0,W_1,…,W_{79}. Radne varijable označavaju se s a, b, c, d, e. Riječi hash vrijednosti označavaju se s $H_0^{(i)}$,$H_1^{(i)}$,…,$H_4^{(i)}$, koje drže početnu hash vrijednost $H^{(0)}$, zamijenjenu svakom uzastopnom hash vrijednosti nakon obrade bloka i završava s konačnom hash vrijednosti $H^{(N)}$. Kod SHA-1 koristi se još i privremena riječ T.

Svi SHA koriste funkcije i konstante pri obradi. Funkcije koje se koriste kod SHA-1 su:

$$f_t(x,y,z) = \begin{cases} Ch(x,y,z) = (x \text{ AND y}) \oplus (\text{NOT x AND z}),\ 0 \le t \le 19 \\ Parity(x,y,z) = x \oplus \text{y} \oplus \text{z},\ 20 \le t \le 39 \\ Maj(x,y,z) = (x \text{ AND y}) \oplus (x \text{ AND z}) \oplus (\text{y AND z}),\ 40 \le \text{t} \le 59 \\ Parity(x,y,z) = x \oplus \text{y} \oplus \text{z},\ 60 \le \text{t} \le 79 \end{cases} \quad (3\text{-}3)$$

gdje su:

- t – broj koraka ili runde ponavljanja, $0 \le t \le 79$
- x, y, z – 32-bitne riječi
- f_t – funkcija koja za tri ulazne 32-bitne riječi vraća 32-bitnu riječ
- *Ch*(x,y,z) – logička operacija kod koje bit x određuje hoće li na izlazu biti bit y ili z
- *Parity*(x,y,z) – logička operacija parnosti, na izlazu vraća bit „1“ ako se radi o neparnom broju bitova „1“ na ulazu
- *Maj*(x,y,z) – logička operacija većine, na izlazu vraća najčešći bit koji se pojavljuje na ulazu

Kod SHA-1 koriste se osamdeset 32-bitnih konstanti K_t, u heksadekadskom sustavu to su:

$$K_t = \begin{cases} 5a827999,\ 0 \le t \le 19 \\ 6ed9eba1,\ 20 \le t \le 39 \\ 8f1bbcdc,\ 40 \le \text{t} \le 59 \\ ca62c1d6,\ 60 \le \text{t} \le 79 \end{cases} \quad (3\text{-}4)$$

Početne hash vrijednosti $H^{(0)}$ za SHA-1 u heksadekadskom zapisu su:

$$H_0^{(0)} = 67452301$$

$$H_1^{(0)} = \text{efcdab89}$$

$$H_2^{(0)} = 98\text{badcfe}$$

$$H_3^{(0)} = 10325476$$

$$H_4^{(0)} = \text{c3d2e1f0}$$

Za SHA-1 svaki blok poruke $M^{(1)}, M^{(2)},\ldots,M^{(N)}$ se redom obrađuje koristeći sljedeće korake:

Za i = 1 do N

{

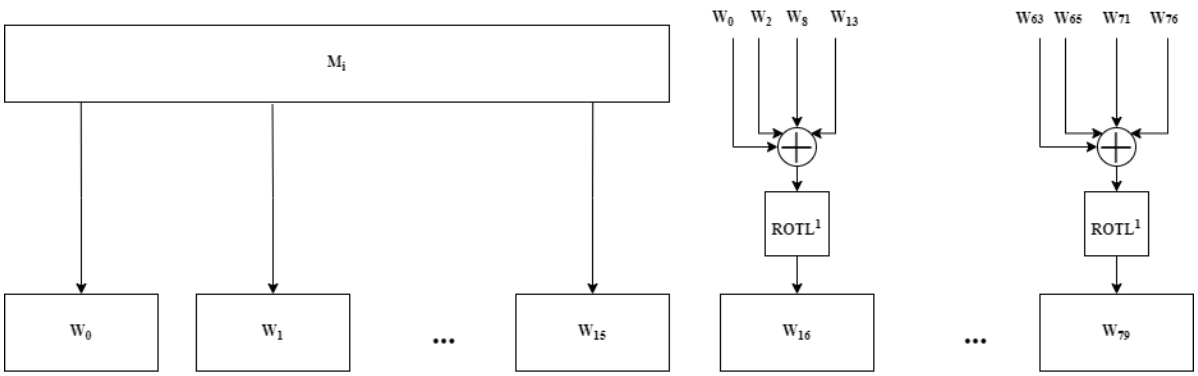
1. Priprema rasporeda poruke $\{W_t\}$:

$$W_t = \begin{cases} M_t^{(i)}, & 0 \le t \le 15 \\ ROTL^1(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}), & 16 \le t \le 79 \end{cases} \quad (3\text{-}5)$$

gdje je:

- W_t – 32-bitna riječ rasporeda poruke
- $M_t^{(i)}$ – riječ bloka poruke i
- $ROTL^n(x)$ – operacija rotacije u lijevo ili kružni pomak u lijevo, gdje je x riječ koja se pomiče a n broj bitova

Proces pripreme rasporeda poruke W_t opisan je slikom 3.2. 512-bitni blok poruke M_i, rastavlja se na šesnaest 32-bitnih riječi te se pri izradi svake sljedeće riječi koriste četiri prijašnje riječi. Tako nastaju osamdeset riječi rasporeda.



Slika 3.2. *Proces stvaranja SHA-1 rasporeda poruke.*

2. Inicijalizacija pet radnih varijabli a, b, c, d, e s (i-1)-tom hash vrijednosti:

   $a = H_0^{(i-1)}$

   $b = H_1^{(i-1)}$

   $c = H_2^{(i-1)}$

   $d = H_3^{(i-1)}$

   $e = H_4^{(i-1)}$

3. Glavna hash operacija:

   Za t = 0 do 79:

   {

   $T = ROTL^5(a) + f_t(b,c,d) + e + K_t + W_t$

e = d

d = c

c = $ROTL^{30}$(b)

b = a

a = T

}

4. Računanje i-te među vrijednosti hash vrijednosti $H^{(i)}$

$H_0^{(i)} = a + H_0^{(i-1)}$

$H_1^{(i)} = b + H_1^{(i-1)}$

$H_2^{(i)} = c + H_2^{(i-1)}$

$H_3^{(i)} = d + H_3^{(i-1)}$

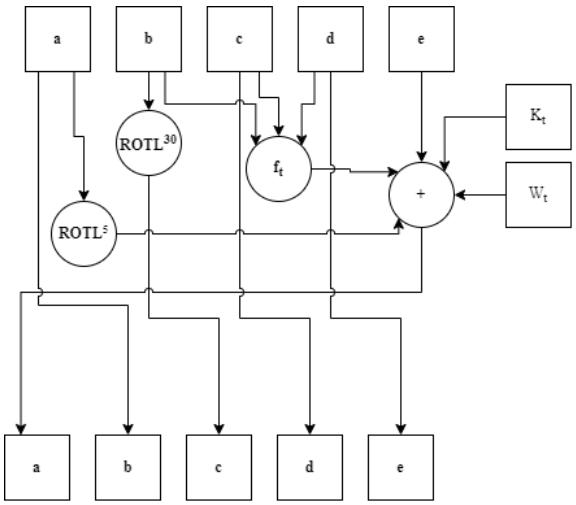
$H_4^{(i)} = e + H_4^{(i-1)}$

}

Nakon ponavljanja ova četiri koraka ukupno N puta, odnosno nakon obrade $M^{(N)}$, rezultat je 160-bitni sažetak poruke M:

$$H_0^{(N)} \| H_1^{(N)} \| H_2^{(N)} \| H_3^{(N)} \| H_4^{(N)}$$

Glavna hash operacija opisana je slikom 3.3. Iz slike se može zaključiti kako izlazne varijable b, d i e nastaju jednostavnom permutacijom varijabli, c jednostavnom rotacijom i permutacijom varijable b, dok je a funkcija svih ulaznih varijabli, konstante K_t i riječi rasporeda W_t.



Slika 3.3. *SHA-1 rotacija.*

3.3. SHA-256

Kao SHA-1, SHA-256 koristi se za stvaranje hash vrijednosti na poruci M, koja je duljine L bita, gdje je $L < 2^{64}$. Razlika je u funkcijama, konstantama, brojevima riječi u rasporedu i hash vrijednosti te broju radnih varijabli. SHA-256 koristi raspored od šezdeset četiri 32-bitne poruke, osam 32-bitnih radnih varijabli te hash vrijednost od osam 32-bitnih riječi te konačni sažetak poruke tada iznosi 256 bitova. Riječi u rasporedu označavaju se s W_0, W_1,…,W_{63}, radne varijable s a, b, c, d, e, f, g i h, a riječi u hash vrijednosti su $H_0^{(i)}$, $H_1^{(i)}$,…,$H_7^{(i)}$ i one drže početnu vrijednost $H^{(0)}$, koja se mijenja svakom uzastopnom hash vrijednosti nakon svake obrade bloka i završava s konačnom hash vrijednosti $H^{(N)}$, kao i kod SHA-1. Još se koriste i dvije privremene riječi T_1 i T_2.

Funkcije koje se koriste za konstruiranje SHA-256 su:

$$\sum_0^{\{256\}}(x) = ROTR^2(x) \oplus ROTR^{13}(x) \oplus ROTR^{22}(x) \tag{3-6}$$

$$\sum_0^{\{256\}}(x) = ROTR^6(x) \oplus ROTR^{11}(x) \oplus ROTR^{25}(x) \tag{3-7}$$

$$\sigma_0^{\{256\}}(x) = ROTR^7(x) \oplus ROTR^{18}(x) \oplus SHR^3(x) \tag{3-8}$$

$$\sigma_1^{\{256\}}(x) = ROTR^{17}(x) \oplus ROTR^{19}(x) \oplus SHR^{10}(x) \tag{3-9}$$

$$Ch(x, y, z) = (x\ AND\ y) \oplus (NOT\ x\ AND\ z) \tag{3-10}$$

$$Maj(x, y, z) = (x\ AND\ y) \oplus (x\ AND\ z) \oplus (y\ AND\ z) \tag{3-11}$$

gdje su:

- $ROTR^n$(x) – operacija rotacije u desno ili kružni pomak u desno, gdje je x riječ koja se pomiče a n broj bitova
- SHR^n(x) – operacija rotacije u desno, popunjena nulama s lijeve strane

Konstante koje se koriste pri konstruiranju SHA-256 napisane u heksadekadskom zapisu nalaze se na slici 3.4. Ima ih šezdeset četiri i dobivaju se iz prva trideset dva bita kubna korijena prva šezdeset četiri prosta broja.

```
428a2f98 71374491 b5c0fbcf e9b5dba5 3956c25b 59f111f1 923f82a4 ab1c5ed5
d807aa98 12835b01 243185be 550c7dc3 72be5d74 80deb1fe 9bdc06a7 c19bf174
e49b69c1 efbe4786 0fc19dc6 240ca1cc 2de92c6f 4a7484aa 5cb0a9dc 76f988da
983e5152 a831c66d b00327c8 bf597fc7 c6e00bf3 d5a79147 06ca6351 14292967
27b70a85 2e1b2138 4d2c6dfc 53380d13 650a7354 766a0abb 81c2c92e 92722c85
a2bfe8a1 a81a664b c24b8b70 c76c51a3 d192e819 d6990624 f40e3585 106aa070
19a4c116 1e376c08 2748774c 34b0bcb5 391c0cb3 4ed8aa4a 5b9cca4f 682e6ff3
748f82ee 78a5636f 84c87814 8cc70208 90befffa a4506ceb bef9a3f7 c67178f2
```

Slika 3.4. *SHA-256 konstante.* [3]

Početne hash vrijednosti $H^{(0)}$ za SHA-256 u heksadekadskom zapisu su:

$$H_0^{(0)} = 6a09e667$$

$$H_1^{(0)} = bb67ae85$$

$$H_2^{(0)} = 3c6ef372$$

$$H_3^{(0)} = a54ff53a$$

$$H_4^{(0)} = 510e527f$$

$$H_5^{(0)} = 9b05688c$$

$$H_6^{(0)} = 1f83d9ab$$

$$H_7^{(0)} = 5be0cd19$$

Svaki blok poruke $M^{(1)}, M^{(2)},\ldots,M^{(N)}$ se redom obrađuje koristeći sljedeće korake:

Za i = 1 do N:

{

1. Priprema rasporeda poruke $\{W_t\}$:

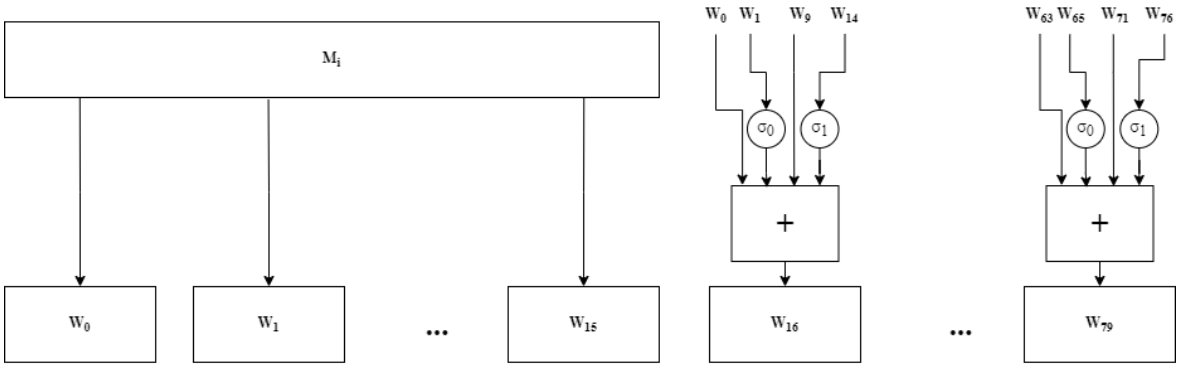
$$W_t = \begin{cases} M_t^{(i)}, & 0 \le t \le 15 \\ \sigma_1^{\{256\}}(W_{t-2}) + W_{t-7} + \sigma_0^{\{256\}}(W_{t-15}) + W_{t-16}, & 16 \le t \le 63 \end{cases} \quad (3\text{-}12)$$

gdje su:

- $\sigma_1^{\{256\}}$, $\sigma_1^{\{256\}}$ – funkcije koje za ulaznu 32-bitnu vrijednost vraćaju 32-bitnu vrijednost

- W_t – 32-bitne riječi rasporeda poruke
- t – broj koraka ili runde ponavljanja, $0 \leq t \leq 63$

Na slici 3.5. nalazi se sustav stvaranja riječi rasporeda poruke za SHA-256. Radi na sličan način kao kod SHA-1. Prvih šesnaest 32-bitnih riječi se dobivaju iz bloka poruke M_i te svaka sljedeća riječ nastaje operacijama nad određenim prethodnim riječima.



Slika 3.5. *Sustav stvaranja SHA-256 rasporeda poruke.*

2. Inicijalizacija osam radnih varijabli a, b, c, d, e, f, g i h s (i-1)-tom hash vrijednosti:

   $a = H_0^{(i-1)}$

   $b = H_1^{(i-1)}$

   $c = H_2^{(i-1)}$

   $d = H_3^{(i-1)}$

   $e = H_4^{(i-1)}$

   $f = H_5^{(i-1)}$

   $g = H_6^{(i-1)}$

   $h = H_7^{(i-1)}$

3. Za t = 0 do 63

   {

   $T_1 = h + \sum_1^{\{256\}}(e) + Ch(e,f,g) + K_t^{\{256\}} + W_t$

   $T_2 = \sum_0^{\{256\}}(a) + Maj(a,b,c)$

   $h = g$

   $g = f$

$f = e$

$e = d + T_1$

$d = c$

$c = b$

$b = a$

$a = T_1 + T_2$

}

4. Računanje i-te među vrijednosti hash vrijednosti $H^{(i)}$

$H_0^{(i)} = a + H_0^{(i-1)}$

$H_1^{(i)} = b + H_1^{(i-1)}$

$H_2^{(i)} = c + H_2^{(i-1)}$

$H_3^{(i)} = d + H_3^{(i-1)}$

$H_4^{(i)} = e + H_4^{(i-1)}$

$H_5^{(i)} = f + H_5^{(i-1)}$

$H_6^{(i)} = g + H_6^{(i-1)}$

$H_7^{(i)} = h + H_7^{(i-1)}$

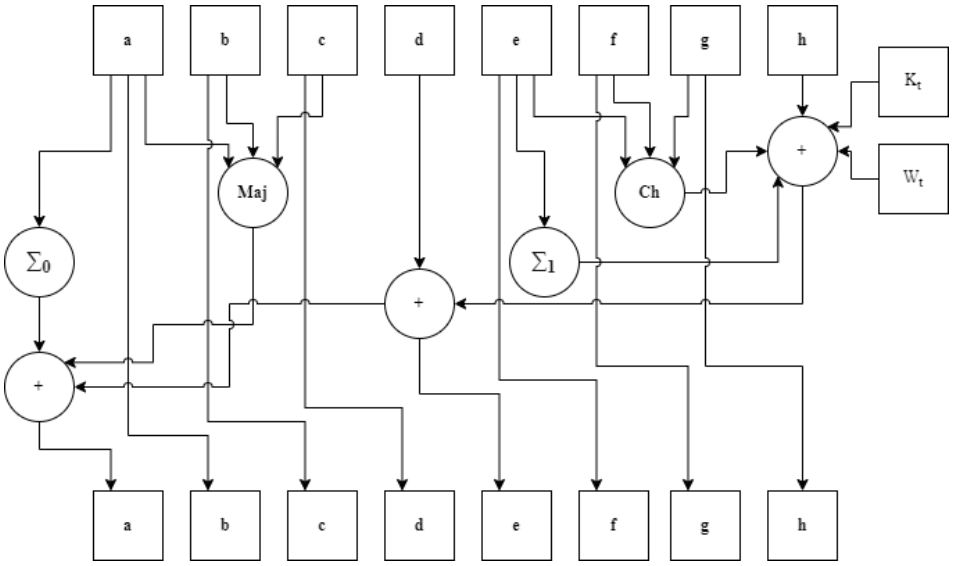
}

Ovaj postupak se ponavlja N puta te kao rezultat ima 256-bitni sažetak dane poruke M u obliku:

$$H_0^{(N)} \| H_1^{(N)} \| H_2^{(N)} \| H_3^{(N)} \| H_4^{(N)} \| H_5^{(N)} \| H_6^{(N)} \| H_7^{(N)}$$

Algoritam SHA-224 radi na isti način. Razlika je u tome što se sažetak poruke algoritma SHA-224 sastoji od sedam 32-bitnih vrijednosti, odnosno sažetak poruke mu je 224-bitan. To se postiže skraćivanjem konačne hash vrijednosti od SHA-256 na krajnja lijeva dvjesto dvadeset četiri bita. Također ima i različite inicijalne hash vrijednosti $H^{(0)}$.

Glavna operacija runde algoritma SHA-256 opisana je slikom 3.6. Može se zaključiti kako su izlazne varijable b, c, d, f, g, h jednostavne permutacije varijabli, dok varijable a i e proizlaze kao rezultat funkcija i operacija nad ostalim ulaznim varijablama.



Slika 3.6. *SHA-256 rotacija.*

3.4. SHA-512

Za razliku od SHA-1 i SHA-256, SHA-512 se koristi za stvaranje hash vrijednosti poruke M, čija duljina L može biti do 2^{128}. Algoritam koristi raspored poruke od osamdeset 64-bitnih riječi, osam radnih varijabli po 64 bita te hash vrijednosti od osam 64-bitnih riječi. Riječi kod rasporeda poruke su označene s W_0, W_1,…, W_{79}, kao i kod SHA-1, a osam radnih varijabli a, b, c, d, e, f, g, h, kao i kod SHA-256. Riječi hash vrijednosti su $H_1^{(i)}$, $H_2^{(i)}$,…,$H_7^{(i)}$. SHA-512 koristi dvije privremene riječi T_1 i T_2.

Funkcije korištene kod SHA-512 slične su onima korištenim kod SHA-256, prilagođenim za 64-bitne vrijednosti, a to su:

$$\sum_0^{\{512\}}(x) = ROTR^{28}(x) \oplus ROTR^{34}(x) \oplus ROTR^{39}(x) \tag{3-13}$$

$$\sum_0^{\{512\}}(x) = ROTR^{14}(x) \oplus ROTR^{18}(x) \oplus ROTR^{41}(x) \tag{3-14}$$

$$\sigma_0^{\{512\}}(x) = ROTR^{1}(x) \oplus ROTR^{8}(x) \oplus SHR^{7}(x) \tag{3-15}$$

$$\sigma_1^{\{512\}}(x) = ROTR^{19}(x) \oplus ROTR^{61}(x) \oplus SHR^{6}(x) \tag{3-16}$$

$$Ch(x,y,z) = (x\ AND\ y) \oplus (NOT\ x\ AND\ z) \tag{3-17}$$

$$Maj(x,y,z) = (x\ AND\ y) \oplus (x\ AND\ z) \oplus (y\ AND\ z) \tag{3-18}$$

Heksadekadski zapis konstanti korištenih u konstruiranju SHA-512 nalazi se na slici 3.7., a to je niz prvih šezdeset četiri bita kubnih korijena od prvih osamdeset prostih brojeva.

```
428a2f98d728ae22 7137449123ef65cd b5c0fbcfec4d3b2f e9b5dba58189dbbc
3956c25bf348b538 59f111f1b605d019 923f82a4af194f9b ab1c5ed5da6d8118
d807aa98a3030242 12835b0145706fbe 243185be4ee4b28c 550c7dc3d5ffb4e2
72be5d74f27b896f 80deb1fe3b1696b1 9bdc06a725c71235 c19bf174cf692694
e49b69c19ef14ad2 efbe4786384f25e3 0fc19dc68b8cd5b5 240ca1cc77ac9c65
2de92c6f592b0275 4a7484aa6ea6e483 5cb0a9dcbd41fbd4 76f988da831153b5
983e5152ee66dfab a831c66d2db43210 b00327c898fb213f bf597fc7beef0ee4
c6e00bf33da88fc2 d5a79147930aa725 06ca6351e003826f 142929670a0e6e70
27b70a8546d22ffc 2e1b21385c26c926 4d2c6dfc5ac42aed 53380d139d95b3df
650a73548baf63de 766a0abb3c77b2a8 81c2c92e47edaee6 92722c851482353b
a2bfe8a14cf10364 a81a664bbc423001 c24b8b70d0f89791 c76c51a30654be30
d192e819d6ef5218 d69906245565a910 f40e35855771202a 106aa07032bbd1b8
19a4c116b8d2d0c8 1e376c085141ab53 2748774cdf8eeb99 34b0bcb5e19b48a8
391c0cb3c5c95a63 4ed8aa4ae3418acb 5b9cca4f7763e373 682e6ff3d6b2b8a3
748f82ee5defb2fc 78a5636f43172f60 84c87814a1f0ab72 8cc702081a6439ec
90befffa23631e28 a4506cebde82bde9 bef9a3f7b2c67915 c67178f2e372532b
ca273eceea26619c d186b8c721c0c207 eada7dd6cde0eb1e f57d4f7fee6ed178
06f067aa72176fba 0a637dc5a2c898a6 113f9804bef90dae 1b710b35131c471b
28db77f523047d84 32caab7b40c72493 3c9ebe0a15c9bebc 431d67c49c100d4c
4cc5d4becb3e42b6 597f299cfc657e2a 5fcb6fab3ad6faec 6c44198c4a475817
```

Slika 3.7. *SHA-512 konstante.* [3]

Početne hash vrijednosti $H^{(0)}$ za SHA-512 su 64-bitne za razliku od SHA-1 i SHA-256, a u heksadekadskom zapisu to su:

$$H_0^{(0)} = 6a09e667f3bcc908$$

$$H_1^{(0)} = bb67ae8584caa73b$$

$$H_2^{(0)} = 3c6ef372fe94f82b$$

$$H_3^{(0)} = a54ff53a5f1d36f1$$

$$H_4^{(0)} = 510e527fade682d1$$

$$H_5^{(0)} = 9b05688c2b3e6c1f$$

$$H_6^{(0)} = 1f83d9abfb41bd6b$$

$$H_7^{(0)} = 5be0cd19137e2179$$

SHA-512 za obradu koristi jednake korake kao i SHA-256. Raspored poruke se stvara od 1024-bitnog bloka poruke. Riječi rasporeda su 64-bitne te zbog toga se koriste funkcije predstavljene jednadžbama (3-15) i (3-16), kao i jednadžbe (3-13) i (3-14) kod glavne hash operacije. Konačni sažetak poruke M je 512-bitni, a sastoji se od osam 64-bitnih riječi.

Algoritmi SHA-384, SHA-512/224 i SHA-512/256 se konstruiraju na isti način kao i SHA-512. Predobrada im je jednaka, uz iznimku početnih hash vrijednosti $H^{(0)}$, a dobivaju se iz SHA-512 uzimajući krajnje lijeve bitove sažetka poruke.

3.5. SHA-3

SHA-3 iako dijeli ime s ostalim algoritmima SHA, predstavlja novu grupu funkcija. Za razliku od ostalih, SHA-3 nije temeljen na MD4, nego na KECCAK algoritmu. Prvi put je objavljen 2015. godine od strane NIST-a pod nazivom SHA-3 *Standard: Permutation-Based Hash And Extendable Output Functions* ili FIPS PUB 202 [4]. Sastoji se od četiri kriptografske hash funkcije SHA-3 sa sufiksima 224, 256, 384 i 512 koji označavaju duljinu sažetka poruke u bitovima i dvije funkcije proširenog izlaza *SHAKE128* i *SHAKE256*. Funkcije proširenog izlaza ili XOF (engl. *extendable-output function*) su funkcije čiji se izlaz može proširiti na željenu duljinu. Sufiksi „128“ i „256“ ne označavaju duljinu sažetka poruke već sigurnosnu razinu.

Funkcije koje čine SHA-3 standard sastoje od konstrukcije spužve (engl. *sponge*). Ta se konstrukcija sastoji od tri dijela: funkcije nad nizom fiksne duljine f, parametrom brzine r (engl. *rate*) i pravilom dopune pad. Skupom tih parametara izrađuju se funkcije spužve. One primaju dva ulazna parametra niz bitova N i duljinu izlazne vrijednosti d, te se označavaju s *SPONGE*[f, pad, r](N, d). Funkcija f preslikava nizove fiksne duljine d u nizove iste duljine. Brzina r je pozitivan broj manji od d, a c je kapacitet koji iznosi d-r. Pravilo dopune pad proizvodi niz odgovarajuće duljine za dodavanje drugom nizu.

KECCAK je skup funkcija spužve. Za temeljnu funkciju permutacije koristi KECCAK-p[b, 12+2l] s pravilom dopune pad10*1. Ovaj skup funkcija ima parametre takve da je $b \in \{25, 50, 100, 200, 400, 800, 1600\}$, s bilo kojom kombinacijom brzine r i kapaciteta c. Za dani niz stanja A i indeks runde i_r funkcija runde *Rnd* obavlja niz preslikavanja θ, ρ, π, χ, and ι, tim redom. KECCAK-p[b, n_r] permutacija sastoji se od n_r iteracija *Rnd*. Funkcija *Rnd*:

$$Rnd(A, i_r) = \iota(\chi(\pi(\rho(\theta(A)))), i_r) \quad (3\text{-}19)\ [4]$$

gdje je:

- θ – operacija A[i][j][k] ← A[i][j][k] ⊕ *Parity*(A[0...4][j−1][k]) ⊕ *Parity*(A[0...4][j+1][k−1]
- ρ – operacija A[i][j][k] ← A[i][j][k−(t+1)(t+2)/2], za 0 ≤ t < 24,
- π – operacija A[3i+2j][i] ← A[i][j]
- χ – operacija A[i][j][k] ← A[i][j][k] ⊕ (*NOT* A[i][j+1][k] *AND* A[i][j+2][k])
- ι – operacija A[0][0][2m−1] *XOR* s bitom m + 7n LFSR (engl. *linear-feedback shift register*) niza osmog stupnja, za 0 ≤ m ≤ l u n rundi

Pseudokod za pravilo dopune *pad10*1* nalazi se na slici 3.8. Kao što se može vidjeti iz koda znak „*“ označava bit „0“ koji je izostavljen ili ponovljen ovisno o potrebi izlazne duljine.

```
Algoritam pad10*1(x, m)

Ulaz:
prirodan broj x;
ne-negativan broj m.

Izlazi:
niz P takav da je m + len(P) pozitivan višekratnik od x

Koraci:
    1. j = (- m - 2) mod x
    2. Vrati P = 1 || 0^j || 1.
```

Slika 3.8. *Pseudokod za pad10*1.*

Kada je funkcija KECCAK ograničena na b=1600, funkcija se označava s KECCAK[c] i njena jednadžba glasi:

$$KECCAK[c](N, d) = SPONGE[KECCAK - p[1600, 24], pad10^*1, 1600 - c](N, d) \quad (3\text{-}20)$$

SHA-3 algoritmi se definiraju iz KECCAK[c] podskupa funkcija. SHA-3 funkcije nalaze se na slici 3.9. Može se vidjeti kako je slučaju kriptografskih funkcija kapacitet dvostruko dulji od sažetka poruke, odnosno c=2d. Kod kriptografskih funkcija na kraj poruke se dodaju bitovi „01“, odnosno N=M||01, a kod funkcija XOF „1111“. To je potrebno za razlikovanja domena ulaza kriptografskih i XOF funkcija.

SHA3-224(M) = KECCAK[448] (M || 01, 224);
SHA3-256(M) = KECCAK[512] (M || 01, 256);
SHA3-384(M) = KECCAK[768] (M || 01, 384);
SHA3-512(M) = KECCAK[1024] (M || 01, 512).

SHAKE128(M, d) = KECCAK[256] (M || 1111, d),
SHAKE256(M, d) = KECCAK[512] (M || 1111, d).

Slika 3.9. *Definicija SHA-3 algoritama.*

3.6. Sigurnosna usporedba SHA algoritama

Tablicom 3.1. uspoređeni su SHA algoritmi na temelju sigurnosti [4]. Sigurnosna jačina u bitovima je broj operacija potrebnih za „razbijanje“ kriptografskih algoritama. Ako je 2^n broj operacija algoritma ili sustava potreban za „razbijanje“ kriptografskog algoritma, onda je njegova sigurnosna jačina u bitovima n. Funkcija L(M) korištena kod napada druge preslike je definirana kao $\lceil \log_2(\text{len}(M)/B \rceil$, gdje je B duljina bloka funkcija. Za SHA-1, SHA-224 i SHA-256 iznosi B=512, dok je B=1024 za SHA-512. Funkcija min je funkcija koja za rješenje ima manji od dva dana broja. SHA-3 kriptografske hash funkcije dizajnirane su da budu alternativa za SHA-2, tako da ispunjavaju jednake sigurnosne potrebe. XOF funkcije SHA-3 pružaju različite razine sigurnosti ovisno o duljini izlazne vrijednosti d.

Tablica 3.1. *Sigurnosna usporedba skupine SHA algoritama.* [4]

Funkcija	**Veličina izlaza**	**Sigurnosna jačina u bitovima**		
		Kolizije	**Preslike**	**Druge preslike**
SHA-1	160	<80	160	160-L(M)
SHA-224	224	112	224	min(224, 256-L(M))
SHA-512/224	224	112	224	224
SHA-256	256	128	256	256-L(M)
SHA-512/256	256	128	256	256
SHA-384	384	192	384	384
SHA-512	512	256	512	512-L(M)
SHA3-224	224	112	224	224
SHA3-256	256	128	256	256

SHA3-384	384	192	384	384
SHA3-512	512	256	512	512
SHAKE128	d	min(d/2, 128)	≥ min(d, 128)	min(d, 128)
SHAKE256	d	min(d/2, 128	≥ min(d, 256)	min(d, 256)

3.7. Protokoli koji koriste SHA

Brojni sigurnosni protokoli i aplikacije koriste SHA. Neki od njih su SSL, TLS, S/MIME i IPsec.

SSL (engl. *Secure Sockets Layer*) je internetski sigurnosni protokol temeljen na enkripciji. Služi za osiguravanje privatnosti, autentičnosti i očuvanje integriteta u internetskoj komunikaciji, što osiguravaju sustavi s kriptografskim hash funkcijama. Razvio ga je Netscape Communications 1994. godine. Kroz godine je bilo više iteracija SSL protokola, a 1999. je unapređen u TLS (engl. *Transport Layer Security*) protokol. TLS je sigurnija verzija SSL-a koja se još uvijek koristi sa zadnjom verzijom TLS 1.3 iz 2018. godine. Internetske stranice koje implementiraju SSL/TLS na mjestu „HTTP“ u URL-u sadrže „HTTPS“.

S/MIME (engl. *Secure/Multipurpose Internet Mail Extensions*) je internetski protokol za slanje digitalno potpisanih i šifriranih poruka. Zadaće tog protokola su potvrda identiteta pošiljatelja i zaštita poruka elektronske pošte. Razvio ga je RSA Security 1995. godine. Za dodatnu sigurnost može se koristiti s drugim protokolima kao na primjer SSL/TLS.

IPsec (engl. *Internet Protocol Security*) predstavlja grupu internetskih protokola koju je definirala organizacija IETF (engl. *The Internet Engineering Task Force*) 1995. godine. Ta grupa služi za uspostavu sigurne komunikacije između dva uređaja. Često se koristi za postavljanje VPN sustava, enkripcijom IP adrese i provjere autentičnosti izvora.

4. IZRADA SHA-256 U MATLABU

MATLAB je programska platforma dizajnirana za inženjere i znanstvenike u svrhu dizajniranja i analize sustava. Kod izrade algoritma SHA-256 potrebno je definirati korištene funkcije. Na slici 4.1. nalaze se funkcije jednadžbi (3-6), (3-7), (3-8) i (3-9). U MATLAB-u se funkcije pišu ključnom riječi *function*, nazivom izlazne varijable koji je u ovim slučajevima *out*, nazivom funkcije s pripadajućim ulaznim parametrima, funkcije su *sum0* za (3-6), *sum1* za (3-7), *sigma0* za (3-8) te *sigma1* za (3-9) s ulaznim parametrom *word* i ključnom riječi *end*, koja označava kraj funkcije. U ovim primjerima korištena je ugrađena funkcija *bitxor* koja na temelju dva predana broja izvršava operaciju isključivo ili (XOR) nad njihovim bitovima. *Bitxor* prima dva parametra, a za ovaj slučaj nam je potrebna operacija nad tri broja. Taj je problem riješen tako da je jedna operacije *bitxor* izvršena nad rješenjem druge i trećem broju koji nije korišten u prvoj operaciji. Za izvršavanje ovih funkcija bile su potrebne i dvije korisnički izrađene funkcije *rotr* i *shr*. Radi se o operacijama kružne rotacije bitova predanog broja te operacije pomaka bitova predanog broja u desno. Obje funkcije primaju dvije vrijednosti, riječ i broj mjesta za koji se ona rotira, odnosno pomiče.

```
function out = sum0( word )
    out = bitxor( bitxor( rotr( word, 2 ), rotr( word, 13 ) ) , rotr( word, 22 ) );
end

function out = sum1( word )
    out = bitxor( bitxor( rotr( word, 6 ), rotr( word, 11 ) ) , rotr( word, 25 ) );
end

function out = sigma0( word )
    out = bitxor( bitxor( rotr( word, 7 ), rotr( word, 18 ) ) , shr( word, 3 ) );
end

function out = sigma1( word )
    out = bitxor( bitxor( rotr( word, 17 ), rotr( word, 19 ) ), shr( word, 10 ) );
end
```

Slika 4.1. *Funkcije algoritma SHA-256.*

Funkcije koje opisuju logičke operacije *maj* i *ch* su na slici 4.2. One imaju tri ulazna parametra x, y, z i nad njima se izvršavaju logičke operacije *bitxor*, *and* i *not*. Te operacije su ugrađene funkcije MATLAB platforme.

```
function out = maj( x, y, z )
    out = bitxor( bitxor( and( x, y), and(x, z )) , and( y, z ));
end

function out = ch( x, y, z )
    out = bitxor( and ( x, y ), and ( not( x ), z ));
end
```

Slika 4.2. *Funkcije maj i ch.*

Na slici 4.3. nalazi se funkcija *mod32add*. S tom funkcijom opisana je operacija zbrajanja 32-bitnih brojeva. Funkcija za ulazni parametar prima varijablu *varargin*. Radi se o ulaznoj varijabli koja omogućuje neodređen broj varijabli ulaza funkciji. Zbrajanje se radi s decimalnim vrijednostima s toga nam je potrebna funkcija *fix2dec* koja prima ulazni niz brojeva i vraća njihovu decimalnu vrijednost. Nakon toga se odvija modularna operacija binarnog zapisa zbroja s 2^{32}. Time se osigurava da je vrijednost 32-bitna. Na kraju se izlaz prebacuje u logički oblik, gdje su sve ne nule zamijenjene s logičkom vrijednosti „1“ i sve nule s logičkom vrijednosti „0“. Kako bi to sve bilo moguće potrebne su ugrađene funkcije *dec2bin*, *mod* i *logical*.

```
function out = mod32add( varargin )
    out = 0;
    for i = 1:length( varargin )
        out = out + fix2dec( varargin{ i } );
    end
    out = dec2bin( mod( ( out ), 2^32 ),32 ) ;
    out = logical( out(:)'-'0' );
end
```

Slika 4.3*. Funkcija mod32add.*

Na slici 4.4. nalazi se funkcija dopune. Funkcija prima poruku, a vraća dopunjenu poruku i njenu duljinu, koja je potrebna za određivanje broja blokova na koje će se rastaviti. Na početku funkcije je deklarirano polje za izlaz koje se postepeno popunjava i definirana je varijabla L koja predstavlja duljinu poruke u bitovima. Polje se popunjava horizontalno binarnim vrijednostima poruke za što se koriste ugrađene funkcije *strcat* i *dec2bin*. Na kraj se dodaje bit „1“ popraćen s k nula, gdje je jednadžba za k dana u (4-1) te je za to potrebna funkcija *mod*. Na kraj dopunjene poruke dodaje se 64-bitna vrijednost duljine poruke. Za to se koristi funkcije *dec2bin* koja duljinu pretvara u 64-bitnu vrijednost i *reshape* koja tu vrijednost preoblikuje u polje s jednim retkom. Konačna dopunjena poruka vraća se u logičkom obliku upotrebom funkcije *logical*.

```
function [p,len] = padd( message )
    padded = [];
    L = length( message ) * 8;
    for i = 1:length( message )
        padded = strcat( padded, dec2bin( message( i ), 8 ));
    end
    padded( end + 1 ) = '1';
    k = mod( 448 - 1 - L , 512 );
    padded( end + 1 : end + k ) = '0';
    padded( end + 1: end + 64 ) = reshape( dec2bin( L, 64 ), 1, [] );
    p = logical(padded(:)-'0');
    len = length( padded );
end
```

Slika 4.4. *Funkcija dopune.*

Funkcija *split2blocks* sa slike 4.5. služi za rastavljanje dopunjene poruke na blokove. Za parametre prima dopunjenu poruku i njenu duljinu, a vraća matricu 512-bitnih blokova, kao i broj blokova.

```
function [out,num_blocks] = split2blocks( p, len )
    num_blocks = len/512;
    out = zeros( num_blocks, 512 );
    for i = 1:num_blocks
        out( i, 1:512 ) = p( ( i-1 )*512 + 1:i*512 );
    end
end
```

Slika 4.5. *Funkcija rastavljanja blokova.*

Funkcija SHA-256 prima poruku kao ulaznu varijablu i vraća 256-bitni sažetak poruke u heksadekadskom zapisu. U njoj su definirane konstante K sa slike 3.4. Na slici 4.6. nalazi se: poziv funkcija za dopunjavanje nad porukom, rastavljanje dopunjene poruke na blokove te definiranje početnih hash vrijednosti za SHA-256. Taj dio predstavlja predobradu. Početne hash vrijednosti su iz heksadekadskog oblika pretvorene u binarni pomoću ugrađene funkcije *hexToBinaryVector*. Na slici je također deklarirano polje rasporeda poruka od šezdeset četiri 32-bitne vrijednosti.

```
[padded_msg,padded_len] = padd( message );
[M,total_blocks] = split2blocks( padded_msg,padded_len );
default_hash = [
                '6a09e667';
                'bb67ae85';
                '3c6ef372';
                'a54ff53a';
                '510e527f';
                '9b05688c';
                '1f83d9ab';
                '5be0cd19'
                ];
H = zeros( 8, 32 );
for j = 1:8
    H(j,:) = hexToBinaryVector( default_hash( j , : ) , 32 );
end
W = zeros( 64, 32 );
```

Slika 4.6. *Predobrada SHA-256 i deklaracija rasporeda.*

Postupak računanje riječi rasporeda poruke nalazi se na slici 4.7. Petljom koja se ponavlja šezdeset četiri puta stvaraju se riječi. Uvjetom se provjerava radi li se o prvih šesnaest ponavljanja te ako se radi poruka se rastavlja na 32-bitne riječi. U suprotnom se rade operacije nad riječima. Ovaj postupak opisan je slikom 3.5.

```
for j = 1:64
    if j >= 1 && j <= 16
        W( j, 1:32 ) = M( i, 32*(j-1)+1:j*32 );
    else
        W( j, 1:32 ) = mod32add(sigma1(W(j-2,:)), W(j-7,:) , sigma0(W(j-15,:)), W(j-16,:));
    end
end
```

Slika 4.7. *Računanje riječi rasporeda poruke W.*

Inicijalizacija radnih varijabli prikazana je na slici 4.8.

```
a = H(1,:);
b = H(2,:);
c = H(3,:);
d = H(4,:);
e = H(5,:);
f = H(6,:);
g = H(7,:);
h = H(8,:);
```

Slika 4.8. *Inicijalizacija radnih varijabli.*

Operacija runde SHA-256 nalazi se na slici 4.9. To je skup operacija koje se izvršavaju u petlji od šezdeset četiri ponavljanja pri čemu se koriste konstante, riječi rasporeda i definirane funkcije. Detaljnije je opisana slikom 3.6.

```
for t = 1:64
    T1 = mod32add(h, sum1(e),ch(e, f, g),hexToBinaryVector(K(t,:),32 ), W(t,:));
    T2 = mod32add( sum0(a), maj( a, b ,c )  );
    h = g;
    g = f;
    f = e;
    e = mod32add( d, T1 );
    d = c;
    c = b;
    b = a;
    a = mod32add( T1, T2 );
end
```

Slika 4.9. *Glavna operacija SHA-256.*

Na slici 4.10. je prikazan je postupak računanja među vrijednosti sažetka poruke.

```
H(1,:) =  mod32add( a, H(1,:)  );
H(2,:) =  mod32add( b, H(2,:)  );
H(3,:) =  mod32add( c, H(3,:)  );
H(4,:) =  mod32add( d, H(4,:)  );
H(5,:) =  mod32add( e, H(5,:)  );
H(6,:) =  mod32add( f, H(6,:)  );
H(7,:) =  mod32add( g, H(7,:)  );
H(8,:) =  mod32add( h, H(8,:)  );
```

Slika 4.10. *Računanje među vrijednosti sažetka poruke.*

Operacije sa slika 4.7., 4.8., 4.9. i 4.10. obavljaju se unutar petlje koja se izvršava za svaki blok poruka.

Ispis konačnog sažetka poruke u heksadekadskom sustavu nalazi se na slici 4.11. Funkcija *horzcat* vraća niz istih vrijednosti u horizontalnom smjeru, a funkcija *lower* velika slova pretvara u pripadajuća mala slova.

```
out = binaryVectorToHex( horzcat( H(1,:), H(2,:), H(3,:), H(4,:), H(5,:), H(6,:) ,H(7,:), H(8,:)  ) );
out = lower(out);
```

***Slika 4.11**. Ispis sažetka poruke u heksadekadskom sustavu.*

Na slici 4.12. prikazan je primjer korištenja SHA-256 funkcije u MATLAB-u. Rezultat je u danom primjeru pohranjen u varijablu *digest*.

```
>> digest = sha256('abc')

digest =

    'ba7816bf8f01cfea414140de5dae2223b00361a396177a9cb410ff61f20015ad'
```

***Slika 4.12.** Primjer korištenja SHA-256 funkcije u MATLAB-u.*

Tablicom 4.1. prikazani su sažetci različitih poruka dobivenih korištenjem algoritma. Kod usporedbe prve i druge poruke može se zaključiti kako zamjena mjesta znakova drastično mijenja sažetak poruke. Podudaranja u znakovima su rijetka i slučajna, kao i kod usporedbe prve i treće poruke. Razlika u prvoj i trećoj poruci je u jednom znaku, što se iz sažetka ne može zaključiti. Četvrta i peta poruka su ponavljanja prve poruke. Kod četvrte poruke radi se o duploj prvoj poruci, a peta poruka je prva poruka ponovljena trideset puta. Iz sažetaka u četvrtoj i petoj poruci se ne može raspoznati ulazna poruka ili da se radi o ponavljanju.

Tablica 4.1. *Primjena algoritma za različite poruke.*

Broj	Poruka	Sažetak poruke
1.	abc	ba7816bf8f01cfea414140de5dae2223b00361a 396177a9cb410ff61f20015ad
2.	acb	8e9766083b3bfc2003f791c9853941b0ea035d1 6379bfec16b72d376e272fa57
3.	sbc	0ebb43b7bc4e92673a4fa933d8c1757688b5ea1 7fe090624f0d6dbcb22dc714a
4.	abcabc	76b99ab4be8521d78b19bcff7d1078aabeb477b d134f404094c92cd39f051c3e
5.	abcabcabcabcabcabcabcabcabcabc abcabcabcabcabcabcabcabcabcabc abcabcabcabcabcabcabc	dc92693dc48fac6684b522b0e01cd7ecc2f00f8 0d024d9822b9985b5ec23fd30

5. ZAKLJUČAK

Kriptografske hash funkcije imaju široko područje primjene. Najčešće se koriste u sustavima koji zahtijevaju očuvanje integriteta poruka u internetskoj komunikaciji. Skupina hash algoritama SHA, osim što zadovoljava sigurnosne zahtjeve kriptografskih hash funkcija, posebno podskupine SHA-2 i SHA-3, jedni su od rijetkih standardiziranih algoritama te vrste.

SHA-1 algoritam i algoritmi podskupine SHA-2 su slični po procesu izrade i načinu rada. Značajna razlika je u korištenim funkcijama i broju radnih varijabli od kojih se sastoje hash među vrijednosti pa i konačna hash vrijednost. Zbog poboljšanih sigurnosnih karakteristika, može se reći kako je SHA-2 direktno unaprjeđenje SHA-1 algoritma.

SHA-3 algoritmi predstavljaju novu grupu funkcija koja se značajno razlikuje od SHA-1 i SHA-2. Nisu razvijeni kako bi unaprijedili skupinu algoritama, već kako bi pružili alternativan oblik stvaranja sažetka poruke. Sigurnosne karakteristike su im jednake i u nekim slučajevima neznačajno bolje od SHA-2.

Iako su na izgled kompliciran, njihova realizacija i primjena je brza i jednostavna.

LITERATURA

[1] W. Stallings, *Cryptography and Network Security Principles and Practice*, Sixth. Pearson, New Jersey, 2014.

[2] S. H. Standard, "FIPS Pub 180-1," *Natl. Inst. Stand. Technol.*, vol. 17, no. 180, p. 15, 1995.

[3] Q. H. Dang, "Secure Hash Standard," *FIBS 180-4 Publ.*, vol. 4, no. August, p. 36, 2015, [Online]. Available: http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf%5Cnhttp://thor.info.uaic.ro/~fltiplea/CC/FIPS180-3.pdf%5Cnhttp://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf

[4] National Institute of Standards and Technology, "FIPS PUB 202 SHA-3 Standard : Permutation-Based Hash and," *NIST Fed. Inf. Process. Stand.*, no. August, 2015.

[5] Rohit, S. Kamra, M. Sharma, and A. Leekha, "Secure hashing algorithms and their comparison," *Proc. 2019 6th Int. Conf. Comput. Sustain. Glob. Dev. INDIACom 2019*, pp. 788–792, 2019.

[6] S. H. Lee and K. W. Shin, "An efficient implementation of SHA processor including three hash algorithms (SHA-512, SHA-512/224, SHA-512/256)," *Int. Conf. Electron. Inf. Commun. ICEIC 2018*, vol. 2018-Janua, pp. 1–4, 2018, doi: 10.23919/ELINFOCOM.2018.8330578.

[7] G. Tsudik, "Message authentication with one-way hash functions," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 22, no. 5, pp. 29–38, 1992, doi: 10.1145/141809.141812.

[8] X. Wang, H. Yu, and Y. L. Yin, "Efficient collision search attacks on SHA-0," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 3621 LNCS, no. 90304009, pp. 1–16, 2006, doi: 10.1007/11535218_1.

SAŽETAK

U ovom radu je bilo potrebno navesti skupinu kriptografskih hash algoritama SHA, objasniti način rada i njihovu primjenu te izraditi algoritam u programskoj platformi MATLAB. U prvom dijelu rada objašnjena je skupina kriptografskih hash funkcija, kao i način rada SHA skupine algoritama. Naveden je proces izrade svakog algoritma te su uspoređeni. Na kraju se nalazi postupak izrade SHA-256 koristeći MATLAB-u.

Ključne riječi: algoritam, integritet, kriptografska hash funkcija, SHA, sigurnost podataka

ABSTRACT

The goal of this thesis was to explain cryptographic hash functions SHA, explain their way of working and their application and to create algorithm using programming platform MATLAB. In the first part of the thesis, a group of cryptographic hash functions is explained, as well as the working method of the SHA group of algorithms. The process of creating each algorithm is listed and they are compared. Finally, there is a procedure for creating SHA-256 using MATLAB.

Keywords: algorithm. integrity, cryptographic hash functions, SHA, data security