UNIVERZITET U BEOGRADU
ELEKTROTEHNIČKI FAKULTET

DIGITALNO POTPISIVANJE IP PAKETA KORIŠĆENJEM BLEJK ALGORITMA ZA HEŠIRANJE

Master rad

Mentor:
doc. dr Zoran Čiča

Kandidat:
Danica Golubičić 2013/3149

Beograd, April 2016.

SADRŽAJ

1. UVOD

U današnje vreme, kada je komunikacija digitalnim putem sve više prisutna, sigurnost komunikacije je od ključne važnosti. Zato su razvijeni različiti mehanizmi zaštite od zlonamernih napada. Neki od mehanizama su zasnovani na heš algoritmima koji imaju široku primenu u informacionim tehnologijama.

Digitalno potpisivanje je mehanizam za utvrđivanje autentičnosti digitalne poruke. Validan digitalni potpis potvrđuje primaocu da je poruka stigla od očekivanog pošiljaoca (autentifikacija) i da poruka nije izmenjena (od neke treće strane) u toku prenosa (integritet). Da li je poruka izmenjena u toku prenosa se utvrđuje poređenjem rezultata heš funkcije koji je određen pre slanja poruke i rezultata izračunatog na prijemu.

Algoritam za digitalno potpisivanje podrazumeva da se na predajnoj strani uz pomoć tajnog ključa od poruke proizvoljne dužine kreira heš rezultat. Zatim se poruka i heš rezultat šalju primaocu. Nakon prijema, primalac koristeći isti tajni ključ, određuje heš rezultat primljene poruke. Ako je taj rezultat isti kao onaj primljen sa porukom, primalac može sa sigurnošću znati da integritet poruke nije ugrožen i da ona nije menjana u toku prenosa.

Hardverska implementacija digitalnog potpisivanja IP paketa je predmet ovog rada. Digitalno potpisivanje je vršeno korišćenjem Blake 256 algoritma. U radu je izvršena verifikacija i analiza performansi navedene implementacije. Implementiran sistem će potencijalno moći da se koristi u mrežnim IP čvorovima.

Hardverska implementacija je realizovana korišćenjem VHDL programskog jezika. Korišćeno je ISE razvojno okruženje za razvoj dizajna za FPGA čipove proizvođača Xilinx. Sama funkcionalna verifikacija bloka za digitalno potpisivanje IP paketa je sprovedena u testbenč okruženju napisanom u VHDL programskom jeziku. Za simulaciju autor je koristio Modelsim 10c, dok je za generisanje ulaznih i očekivanih izlaznih podataka koristio referentni softverski model napisan u MATLAB softverskom okruženju.

Ostatak rada je organizovan na sledeći način. Drugo poglavlje daje osnovne informacije o heš funkcijama i digitalnom potpisu. Treće poglavlje opisuje strukturu IP paketa, kao i način računanja kontrolne sume (eng. *check sum*). Četvrto poglavlje opisuje osnovne karakteristike Blake 256 heš funkcije korišćene prilikom digitalnog potpisa. Peto poglavlje detaljno opisuje strukturu sistema i njegovu implementaciju, a u šestom poglavlju je prikazana verifikacija dizajna i dat pregled performansi za urađenu FPGA implementaciju.

2. HEŠ FUNKCIJE

Heš funkcija je izračunljiva funkcija koja, primenjena na poruku promenljive dužine, daje njenu reprezentaciju fiksne dužine koja se naziva njenom heš vrednošću. Heš vrednost je najčešće znatno manja od same poruke i generisana je heš algoritmom na takav način da je verovatnoća da neka druga poruka ima istu heš vrednost zanemarljiva. Ta osobina heš funkcije se naziva i „koliziona otpornost" (eng. *collision resistance*). Druga važna osobina svake heš funkcije je njena jednosmernost što znači da se za zadatu izlaznu vrednost heš funkcije teško može naći odgovarajuća ulazna poruka. Heš funkcija treba da zadovolji tri osnovne karakteristike sigurnosti informacija: poverljivost, integritet i autentifikaciju. Karakteristika poverljivosti omogućava samo primaocu da pročita poslatu poruku. Karakteristika integriteta poruke definiše njenu otpornost na promenu njenog sadržaja pre nego što stigne do primaoca. Autentifikacija predstavlja proces utvrđivanja identiteta pošiljaoca.

Sigurnost današnje komunikacije se dobrim delom zasniva na kriptografskim protokolima, gde mnoštvo takvih protokola koristi heš funkcije kao gradivne blokove. Uloga kriptografskih heš funkcija je nezaobilazna u aplikacijama za digitalni potpis. Da bi se heš funkcija koristila u takvim aplikacijama mora zadovoljavati rigorozne sigurnosne kriterijume. Stvaranje kriptografskih heš funkcija sa dobrim sigurnosnim karakteristikama je važan i istovremeno težak zadatak. Od mnoštva ponuđenih konstrukcija samo su retke preživele probu vremena i pokazale zadovoljavajuće osobine. Danas se među najpoznatijim i najčešće korišćenim heš algoritmima smatraju MD5, SHA-1, SHA-2 i SHA-3. Kriptografski Blake heš algoritam je bio jedan od finalista poslednje treće runde na takmičenju za novi SHA-3 standard koje se završilo u novembru 2012. godine. Iako je pobednik bio Keccak heš algoritam, Blake se smatra kao veoma snažna heš funkcija koja ima veliku sigurnosnu marginu i veoma dobre performanse, a takođe je podobna i za softversku i za hardversku implementciju.

Strogo govoreći, razlikuju se dve vrste heš funkcija, sa jednim i sa dva ulaza. Za heš funkcije sa jednim ulazom, koristi se naziv *manipulacioni detekcioni kod* (MDC - *Manipulation Detection Code*). Blake heš funkcija je primer heš funkcije sa jednim ulazom (*single-input*). Drugi tip heš funkcija ima dva ulaza, od kojih je jedan ulaz poruka, a drugi je tajni ključ. Ove heš funkcije se zovu *heš funkcije sa ključem* (*keyed hash functions*), čiji je najznačajniji predstavnik MAC (*Message Authentication Code*). Očigledno, i jednoulazne heš funkcije mogu imati ulogu heš funkcije sa ključem, tako što se ključ spoji sa porukom i tako formira jedan ulaz.

2.1. Digitalni potpis

Digitalni potpisi su standardni element većine kriptografskih protokola, a najčešće se koriste za distribuciju softvera, u finansijskim transakcijama, i u drugim slučajevima u kojima je važno da se otkrije falsifikovanje i manipulacija.

Kriptografija sa javnim ključevima omogućava da bilo koja poruka koju šalje bilo koji korisnik sadrži digitalni potpis, analogan uobičajenom potpisu u papirnoj korespondenciji. Mogućnost digitalnog potpisivanja omogućava korisniku na prijemnoj strani da se uveri da mu je

poruku poslao legitiman pošiljalac. Sa druge strane, digitalni potpis daje veću garanciju od običnog potpisa da primljeni dokument nije modifikovan.

Digitalni potpisi se klasifikuju na različite načine:

- Implicitni – ako se nalaze u samoj poruci.
- Eksplicitni – ako su dodati uz poruku, kao neodvojivi deo.
- Privatni – ako ga može identifikovati jedino neko ko poseduje zajedničku tajnu informaciju sa pošiljaocem.
- Javni (ili istinski) – ako bilo ko može da identifikuje pošiljaoca na osnovu javno dostupne informacije.
- Revokabilni – ako pošiljalac može kasnije da negira da digitalni potpis pripada njemu.
- Irevokabilni – ako primalac može da dokaže da je pošiljalac autor poruke.

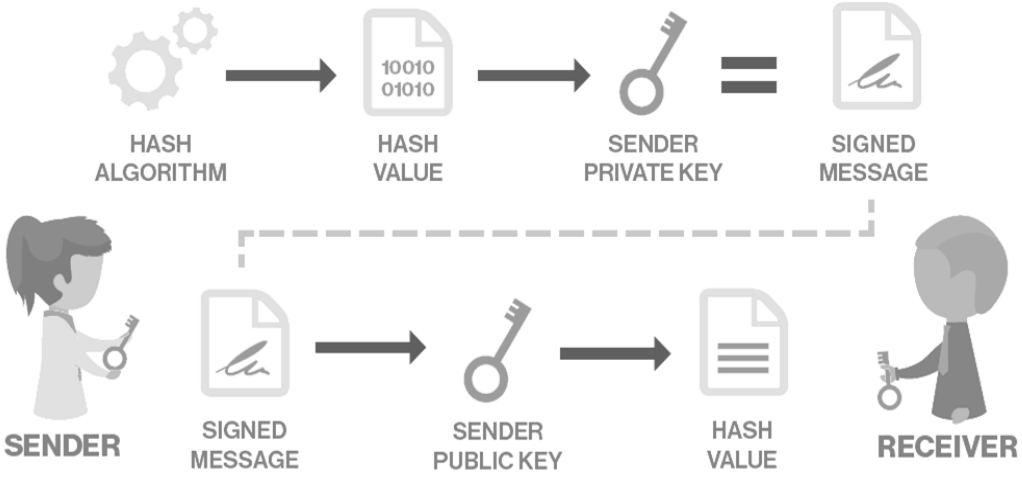
Digitalni potpisi moraju se jednostavno kreirati i verifikovati, a teško falsifikovati. Proces digitalnog potpisivanja jedne poruke sastoji se od dva dela: najpre se računa potpis korisnika koji odgovara poruci, koji samo korisnik može generisati na osnovu svog privatnog ključa i poruke koju želi da potpiše, a zatim se šifruje potpis i šalje se javnim kanalom.

2.1.1. Uopšteno o nastanku digitalnog potpisa

Temelji za pouzdanu proveru porekla informacija, «digitalni potpis», stvoreni su 1976. godine otkrićem kriptografije javnog ključa (Diffie-Hellman), koja se još naziva i asimetričnom kriptografijom. Zanimljivo je napomenuti da je ovaj način kriptovanja podataka, prema nekim informacijama bio poznat britanskoj tajnoj službi nekoliko godina pre nego spomenutoj dvojici istraživača.

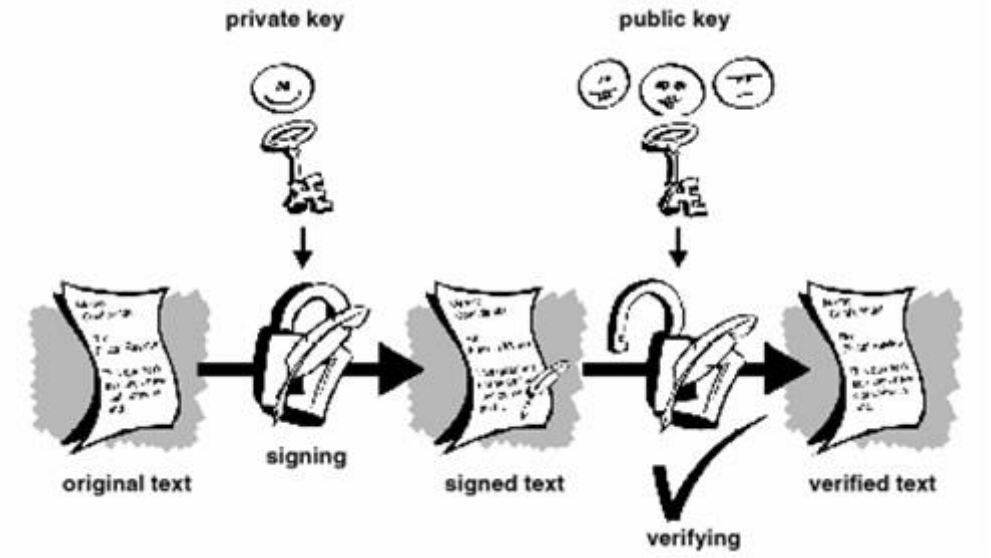
Danas, kada većina razvijenih zemalja u svoje zakone uvodi i zakon o digitalnom potpisu, ovo područje se nalazi na granici dva sveta, kriptografije i prava. Osim pravnih problema oko primene digitalnog potpisa, postoje i pravni problemi vezani za implementaciju algoritama digitalnog potpisa, uglavnom zbog softverskih patenata kojima je velik broj algoritama zaštićen, ali i zbog restriktivnih regulativa pojedinih zemalja vezanih uz kriptografske proizvode. Tako je npr. izvoz «jakog» enkripcijskog softvera iz SAD-a bio zabranjen sve do pred kraj 1999. godine. Isto tako, u Francuskoj je upotreba alata za enkripciju bila zabranjena do početka 1999. Ipak, naglim širenjem elektronskog poslovanja postalo je nužno ovakve odredbe ukinuti, i omogućiti kako sigurnu zaštitu informacija šifriranjem tako i zaštitu od mogućih prevara, autentifikacijom. Kao rezultat napora da se reše navedeni problemi razvijena je tehnika digitalnog potpisa.

2.1.2. Osnovni principi rada digitalnog potpisivanja



Slika 2.1.1. Princip digitalnog potpisa [1]

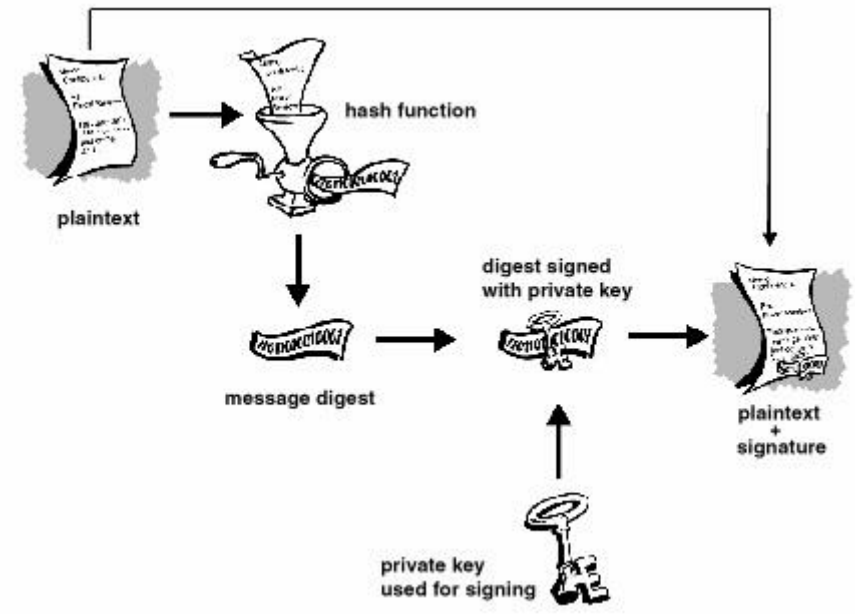
"Digital Signature Algorithm" se koristi za potpisivanje kao i za proveru valjanosti već načinjenog potpisa. Svaki potpisnik poseduje javni i tajni ključ kao što je prikazano na slici 2.1.1. Tajni ključ se koristi u procesu stvaranja digitalnog potpisa, a javni ključ se koristi u procesu provere valjanosti potpisa. U oba slučaja se algoritam primjenjuje na sažetak poruke, *M*, načinjen upotrebom "Secure Hash Algorithm" funkcije. Bez poznavanja tajnog ključa potpisnika nije moguće stvoriti valjan potpis, tj. nije moguće falsifikovati potpis (u razumnom vremenskom periodu). Shodno tome, svako je u mogućnosti proveriti valjanost potpisa korišćenjem javnog ključa potpisnika.



Slika 2.1.2. Princip digitalnog potpisa [2]

Pretpostavimo da dvoje ljudi A i B, žele razmenjivati potpisane poruke (podatke) tj. žele biti sigurni u identitet osobe od koje su poruku dobili (slika 2.1.2.). Kao prvo, obe osobe kreiraju par komplementarnih ključeva, javni i tajni ključ. Važno je naglasiti da se poznavanjem javnog ključa ne može izračunati tajni ključ u nekom razumnom vremenu (vreme potrebno za izračunavanje

tajnog ključa iz poznatog javnog ključa, tj. razbijanje šifre, meri se milionima godina na danas najjačim raspoloživim računarima). Nakon kreiranja ključeva, osobe A i B razmenjuju svoje javne ključeve, a potom pošiljalac (A), koristi svoj tajni ključ za šifrovanje sažetka poruke koji je izračunao nekom od heš funkcija (u ovom radu je korišćena Blake-256 o kojoj će kasnije biti reči). Heš funkcija je funkcija koja iz zadate poruke (podataka) računa sažetak fiksne dužine (u našem slučaju 256 bita), obično od 128 do 256 bita. Kada primalac (B) uspe dešifrovati sažetak poruke javnim ključem pošiljaoca (A), on još računa i sažetak primljene poruke koji potom upoređuje s upravo dešifrovanim, i ako je izračunati sažetak jednak onom dešifrovanom, primalac može biti siguran u poreklo poruke (podataka), jer je poruka mogla biti šifrovana jedino tajnim ključem pošiljaoca (A), kao i u integritet poruke (slika 2.1.3.).



Slika 2.1.3. Princip potpisivanja poruke

U celoj proceduri samo je jedna stvar slaba karika. Moramo biti apsolutno sigurni da javni ključ za koji mislimo da pripada pošiljaocu (A) zaista i pripada pošiljaocu (A). Naime, ukoliko primalac (B) ima javni ključ pošiljaoca (C), a veruje da ključ pripada pošiljaocu (A), tad je pošiljalac (C) u mogućnosti zloupotrebiti podatke pošiljaoca (A).

2.1.3. Potpisi i zakonske regulative

Zakonske regulative (zemalja koje imaju zakon o digitalnom potpisu) ne određuju niti jednu tehnologiju potpisivanja kao dominantnu, već samo donose propise kojih se svaka od tehnologija mora pridržavati. Kao prvo, od digitalnog se potpisa očekuje da bude jedinstven osobi koja ga koristi, drugo, da se može proveriti kome pripada odnosno da li zaista pripada osobi koja ga je koristila, treće, da je u potpunoj kontroli osobe koja ga koristi, četvrto, da potvrđuje i sebe i podatke koje potpisuje.

Već iz ovog vidimo da postoji znatna prednost digitalnog potpisa nad klasičnim metodama autentifikacije. Najveća prednost je ta što se valjanost potpisa proverava svaki put pri primanju dokumenta, za razliku od klasičnih potpisa koji se proveravaju tek na sudu, kad se prevara već odigrala. Osim ove prednosti postoji još jedna značajna prednost, a to je nemogućnost naknadne izmene potpisanog dokumenta, kao i nemogućnost potpisivanja praznih dokumenata. Ipak, ukoliko

falsifikator uspe doći do tajnog ključa, tada on bez ikakvih problema može falsifikovati podatke bez da postoji i najmanja mogućnost utvrđivanja različitosti takvog potpisa od pravog potpisa, što kod klasičnih metoda ipak nije slučaj.

Nemali broj kriptografskih algoritama zaštićen je različitim patentima. Tako je npr. najraširenији asimetrični kriptografski algoritam, RSA, bio patentiran 17 godina sve do septembra 2000. godine, a drugom isto tako vrlo značajnom protokolu, Diffie-Hellman protokolu, patent je istekao u aprilu 1997. godine. Nažalost, ovo nisu jedini takvi primeri kao ni jedini problemi u široj primeni kriptografskih algoritama. Proizvođači kriptografskog softvera (bili) su prisiljeni da proizvode po dve ili čak tri različite verzije istog softverskog paketa da bi se udovoljilo svim izvoznim i patentnim regulativama; zabrana izvoza kriptografskog softvera iz SAD-a s ključem dužim od 40-bita bila je na snazi do pred kraj 1999. godine, tako da je npr. NAI («Network Associates») bio prisiljen imati dve verzije svog programa PGP, jednu za tržište SAD-a a drugu za internacionalno tržište. Ipak, obe su verzije koristile «jaku» enkripciju tako što je iskorišćena «rupa» u zakonu SAD-a kojim se zabranjuje izvoz softvera u binarnom obliku ali ne i izvornog koda na papiru. Osim ovog, postojao je i problem zbog patentiranog RSA algoritma, tako da verzije za SAD nisu koristile istu varijantu kao i internacionalne.

Sličan problem javlja se i s pojavom novog američkog standarda za digitalno potpisivivanje («Digital Signature Standard», DSS, 1994-te), jer se delovi korišćenog algoritma (koji sam po sebi nije patentiran) nalaze pod patentnom zaštitom autora sličnog algoritma temeljenog na diskretnim logaritmima. Ovaj primer ipak odskače iz mnoštva drugih upravo zbog toga što se radi o standardu koji su obavezne koristiti sve državne ustanove SAD-a; drugim rečima ovaj će patent vrlo verovatno američki poreski obveznici osetiti na vlastitom džepu.

Na kraju spomenemo još i zabranu izvoza bilo kakvog kriptografskog softvera zemljama poput Iraka, Severne Koreje ili Kube.

2.1.4. Izazovi i mogućnosti

Ukoliko se pristupi primeni sistema digitalnih potpisa u trgovini, to sa sobom nosi određene prednosti, ali i određene troškove. Troškovi se sastoje od sledećeg:

- Troškovi režije – da bi se sistem izgradio, određena suma novca se plaća ovlašćenima za verifikaciju, korišćenje i skladištenje informacija (kao i drugih usluga vezanih za prethodne aktivnosti).
- Troškovi pretplate – korisniku će biti potreban softver, i moraće da plati ovlašćenoj agenciji da bi dobio potvrdu.
- Takođe se preporučuje hardver koji osigurava privatnu šifru korisnika.
- Osobe koje koriste digitalni potpis će takođe snositi troškove za verifikaciju softvera kao i pristup sertifikatu i listama poništenih sertifikata.

2.1.5. Prednosti sistema

Ukoliko se pravilno ugrade i koriste, digitalni potpisi nude rešenje za sledeće probleme:

- Prevare - svodeći na minimum rizik od poslovanja sa osobama koje žele da izbegnu odgovornost tvrdeći da se neko lažno predstavljao umesto njih.
- Obezbeđuje verodostojnost poruke, jer isključuje rizik da će se određena poruka falsifikovati, i negira lažne tvrdnje da je poruka izmenjena nakon slanja.

- Rešava problem formalno pravnih zahteva, potvrđuje stav da su pravni zahtevi, kao što su pisana forma, potpis i originalni document, zadovoljeni, jer digitalni potpisi funkcionišu isto tako dobro kao i oni na papiru, čak i bolje.
- Problem dostupnosti drugim sistemima zato što štiti informaciju čak iako se ona šalje kanalima koji su otvoreni, nesigurni, i zbog toga jeftini i masovno korišćeni.

3. STRUKTURA IP PAKETA (IPV4 PROTOKOL)

Internet protokol verzija 4 (IPv4) i dalje predstavlja dominantnu verziju IP protokola na Internetu. IP protokol omogućava usmeravanje IP paketa kroz Internet mrežu između polaznog i odredišnog mrežnog uređaja na osnovu IP adrese u zaglavljima tih paketa. IP paket se sastoji od dela zaglavlja i dela podataka. IP paket nema kontrolnu sumu podataka ili bilo koje polje za verifikaciju ispravnog prenosa kompletnog IP paketa zato što se ta provera već obavlja na nižem sloju prenosa.

3.1. Zaglavlje IP paketa

Zaglavlje IPv4 paketa se sastoji od 14 polja, od kojih su 13 obavezna. Poslednje polje u zaglavlju je opciono.

Tabela 3.1.1. IP zaglavlje

<table>
<tr><th>Bitovi</th><th>0-3</th><th>4-7</th><th>8-15</th><th>16-18</th><th>19-31</th></tr>
<tr><td>0-31</td><td>Version</td><td>IHL</td><td>Type of Service</td><td colspan="2">Total length</td></tr>
<tr><td>32-63</td><td colspan="3">Identification</td><td>Flags</td><td>Fragment Offset</td></tr>
<tr><td>64-95</td><td colspan="2">Time To Live</td><td>Protocol</td><td colspan="2">Header Checksum</td></tr>
<tr><td>96-127</td><td colspan="5">Izvorišna adresa</td></tr>
<tr><td>128-159</td><td colspan="5">Odredišna adresa</td></tr>
</table>

Izgled IP zaglavlja prikazan je u tabeli 3.1.1, a detaljniji opis polja IP zaglavlja dat je u nastavku:

- ***Version*** – Verzija IP protokola dužine 4 bita. Ovo polje ima vrednost 4 za IPv4.
- ***IHL* (*Internet Header Length*)** – Dužina IP zaglavlja izražena u broju 32-bitnih reči. Pošto zaglavlje može da sadrži različit broj opcionih polja, u ovom polju navedena je dužina zaglavlja. Minimalna vrednost ovog polja je 5, što predstavlja dužinu od 160 bita ili 20 bajtova. Pošto je u pitanju četvorobitno polje, maksimalna dužina zaglavlja je 15 reči, odnosno 480 bita ili 60 bajtova.
- ***Type of Service*** – ovo polje je 8-bitno i oblika je PPPDTRXX, gde PPP definiše prioritet datagrama (od 0 do 7), D postavlja mali iznos kašnjenja za datu uslugu, T definiše visoku propusnost, R postavlja visoku pouzdanost, a XX su rezervisani biti.

- ***Total length*** – Ukupna dužina čitavog paketa u bajtovima. Minimalna dužina paketa je 20 bajtova (takav paket sadrži samo najkraće zaglavlje dužine 20B) a maksimalna 65,535 bajtova jer je to najveća vrednost koja se može predstaviti pomoću 16 bita.
- ***Identification*** – Niz brojeva koji zajedno sa izvorišnom adresom, odredišnom adresom i korisničkim protokolom treba da jedinstveno identifikuje paket.
- ***Flags*** – sastoji se od tri bita koja se koriste u procesu fragmentacije.
- ***Fragment Offset*** – koristi se u procesu fragmentacije.
- ***Time To Live (TTL)*** - u praksi TTL polje predstavlja maksimalni broj hopova i svaki ruter dekrementira za 1 to polje.
- ***Protocol*** – u datagramu mogu biti enkapsulirani različiti protokoli (TCP, UDP, SCTP,...) i ovo polje identifikuje protokol koji je enkapsuliran u datagram.
- ***Header Checksum*** – ovo polje sadrži 16-bitni bit oblik na osnovu koga se može detektovati greška u prenosu. Ako se prilikom verifikacije kontrolne sume utvrdi da u zaglavlju postoji greška, taj paket se odbacuje. U slučaju promene bilo kog polja zaglavlja potrebno je izračunati novu kontrolnu sumu i ažurirati polje *Header Checksum.*
- ***Source and destination IP addresses*** - izvorišna i odredišna IP adresa smešta se u 32-bitnom polju izvorišne i odredišne IP adrese, respektivno.

3.2. IPv4 kontrolna suma

IPv4 kontrolna suma je jednostavna kontrolna suma korišćena u verziji 4 internet protokola (IPv4). Svrha mu je da zaštiti zaglavlje podataka IPv4 paketa od korupcije podataka. Ova kontrolna suma se računa samo za bajtove zaglavlja (sa vrednošću kontrolne sume bajtova podešene na 0), dugačka je 16 bitova i deo je IP paketa zaglavlja.

Računa se formiranjem odgovarajućeg komplementa (prevrtanje bita 0 u bit 1 i obrnuto) odgovarajućih komplemenata sume 16-bitnih reči zaglavlja. Rezultat sumiranja čitavog IP zaglavlja, uključujući i kontrolnu sumu, trebalo bi da bude 0, osim ako nema oštećenja podataka. Ruter mora prilagoditi kontrolnu sumu ako menja neki deo IP zaglavlja (npr prilikom dekrementiranja TTL-a).

IPv6 protokolu fali polje kontrolne sume zaglavlja. Njegovi kreatori su smatrali da je suvišno praviti odvojeno polje kontrolne sume zaglavlja ako je kontrolna suma celog paketa dostupna sloju linka za podatke i drugom sloju transporta kao PPP i Ethernet i ako se kontrolna suma celog paketa kombinuje sa korišćenjem kontrolne sume u protokolima višeg sloja kao što su TCP i UDP.

3.2.1. Primer računanja IPv4 kontrolne sume

Uzmimo sledeći skraćeni isečak IPv4 paketa kao primer. Zaglavlje je boldovano, a kontrolna suma je podvučena.

4500 0073 0000 4000 4011 b861 c0a8 0001
c0a8 00c7 0035 e97c 005f 279f 1e4b 8180

Za računanje kontrolne sume, prvo računamo sumu svake 16-bitne vrednosti u zaglavlju osim same kontrolne sume (podvučena je u gornjem primeru). Vrednosti su prikazane u heksadecimalnom zapisu.

4500 + 0073 + 0000 + 4000 + 4011 + c0a8 + 0001 + c0a8 + 00c7 = 2479C (ili 149404 u decimalnom zapisu).

Sledeća je konverzija vrednosti 2479C u binarni zapis:

0010 0100 0111 1001 1100

Prva četiri bita su nosioci sume i biće dodata ostatku vrednosti:

0010 + 0100 0111 1001 1100 = 0100 0111 1001 1110

U ovom primeru dodavanje nosioca sumi nije prouzrokovalo stvaranje novog nosioca sume. Da se to kojim slučajem desilo, bilo bi neophodno uvrstiti i tog novog nosioca sume.

Da bi dobili kontrolnu sumu potrebno je da obrnemo bite u novonastalom nizu. (0 prebacujemo u 1 i obrnuto).

0100 0111 1001 1110 postaje:

1011 1000 0110 0001

što je u heksadecimalnom zapisu b861 tj početna kontrolna suma (podvučena gore).

3.2.2. Primer verifikacije IPv4 kontrolne sume

Prilikom verifikacije kontrolne sume, koristi se ista procedura kao u prethodnom primeru, samo što sad originalnu kontrolnu sumu ne izostavljamo.

Primer:

4500 + 0073 + 0000 + 4000 + 4011 + b861 + c0a8 + 0001 + c0a8 + 00c7 = 2fffd

Dodavanjem nosioca sume ostatku vrednosti dobija se:

fffd + 2 = ffff

Formiranjem odgovarajućeg komplementa (obrtanjem svakog bita) dobija se 0000, što znači da zaglavlje datagrama ne sadrži bitske greške. IP kontrolna suma ne proverava redosled 16-bitnih vrednosti u zaglavlju.

4. BLAKE 256

Blake je kriptografska heš funkcija koju su napravili Žan Filip Omason, Luka Henzen, Vili Meier i Rafael Fan. Prototip funkcije je napravljen oktobra 2008, a finalna verzija januara 2011. godine. Blake predstavlja familiju od 4 heš funkcije: Blake-224, Blake-256, Blake-384 i Blake-512 čija je osnovna razlika u dužini izlazne heš vrednosti. U ovom radu za digitalno potpisivanje IP paketa koristi se Blake-256 heš funkcija.

Blake 256 algoritam kao ulaz prihvata poruku maksimalne dužine 2^{64} bita. Takva poruka se najpre dopunjuje (eng. *padding*) dodatnim bitima do poruke koju je moguće podeliti na celobrojni broj 512-bitnih blokova. Zatim se navedeni blokovi podataka dovode redom kao ulaz u kompresionu funkciju. Pored ulaznog bloka podatka u kompresionu funkciju se dovodi i heš rezultat dobijen iz prethodnog bloka. Standard definiše inicijalnu heš vrednost koja figurira prilikom prvog poziva kompresione funkcije. Heš vrednost koja je dobijena obradom poslednjeg ulaznog bloka u nizu uzima se kao konačna heš vrednost cele poruke.

4.1. Kompresiona funkcija

Blake 256 kompresiona funkcija kao ulaz uzima 4 vrednosti:

- Niz trenutne heš vrednosti $h = h_0,.., h_7$
- Blok ulazne poruke $m = m_0,...,m_{15}$
- Salt $s = s_0,...,s_3$
- Brojač $t = t_0,t_1$

Ova četiri ulaza predstavljena su sa ukupno 30 reči dužine po 32 bita. Izlaz ove funkcije je nova heš vrednost $h' = h_0',.., h_7'$ dužine 256 bita.

Slika 4.1.1. Dijagram Blake 256 kompresione funkcije

Dijagram Blake 256 kompresione funkcije sa prikazanim ulaznim i izlaznim vrednostima prikazan je na slici 4.1.1.

Brojač t predstavlja koristan broj bita koji je do tog trenutka obrađen. On obezbeđuje da identični blokovi na različitim mestima u ulaznoj poruci vraćaju različitu heš vrednost i omogućuje viši nivo sigurnosti heš funkcije.

Salt vrednost povećava sigurnost jer korisnik na taj način može da za isti podatak napravi više različitih heš vrednosti. Podrazumevana (eng. *default*) vrednost salta je nula. Salt je koristan za nasumično heširanje.

Kompresiona funkcija čiji su ulazi *h*, *m*, *s*, *t*, a izlaz h' se predstavlja izrazom:

$$h' = compress(h,m,s,t)$$

Autor je prilikom implementacije bloka za digitalno potpisivanje IP paketa koristio već implementiranu kompresionu funkciju [3]. U tom radu se mogu pronaći detalji Blake 256 algoritma.

4.2. Dopuna (*padding*)

Poruka se dopunjava pre heširanja iz mnogo razloga. Prvenstveno se dopunjava da bi mogla biti podeljena u celobrojan broj blokova dužine 512 bita. Povećavanjem dužine poruke, logično menja se i informacija koja će biti prerađena Na primer, ako poruka (informacija) ima 592 bita, bez dopune, prvih 512 bita bi mogli biti smešteni u blok, ali onda će ostalih 80 bita biti neiskorišćeni. Da bi se rešio ovakav problem, pribegava se dopunjavanju poruke do 1024 bita. Ovakvu poruku možemo podeliti u dva bloka od po 512 bita.

Generisanje dopune se može grubo podeliti u tri koraka:

- U prvom koraku se ulazna informacija dopunjava bitom vrednosti jedan praćenim određenim brojem bitova vrednosti nula. Novonastala poruka ima dužinu čija vrednost pri deljenju sa 512 ima ostatak 447. Najmanji broj bita koji u ovom koraku može biti dodat je dva (bit jedan praćen bitom nula), a maksimalan broj bita iznosi 513 (bit jedan pracen nizom od 512 bitova vrednosti nula).
- U drugom koraku se na postojeću dopunu dodaje bit vrednosti jedan.
- U trećem koraku se na postojeću dopunu dodaje 64-bitna unsigned big-endian reprezentacija ukupnog broja bita ulazne poruke.

Iz izloženog se lako može zaključiti da minimalan broj bita dopune iznosi 67 bita, a maksimalan broj bita dopune iznosi 578 bita.

U nastavku će biti pokazano računanje dopune za dužinu ulazne poruke od 1000 i 800 bita.

- U prvom koraku se generiše niz dužine 471 bita pri čemu prvi bit ima vrednost jedinice, a ostali biti imaju vrednost nule. Na ovaj način dužina novoformirane poruke iznosi 1471 bita. Ostatak pri deljenju vrednosti 1471 sa 512 iznosi 447 pri čemu je zadovoljen uslov prvog koraka.
- U drugom koraku se dodaje bit jedan.
- U poslednjem koraku se dodaju 64 bita vrednosti “000….0001111101000” koji kad se pretvore u decimalni oblik čine vrednost 1000 odnosno dužinu poruke.

Spajajući bite dopune iz sva tri koraka dobija se ukupna dužina dopune od 536 bita.

Analogno prethodno opisanom postupku se dobija i dopuna ulazne poruke od 800 bita.

- U prvom koraku se generiše niz dužine 159 bita tako da dužina novoformirane poruke pri deljenju sa 512 daje ostatak 447. Vrednost prvog elementa niza je jedinica, dok su ostali biti jednaki nuli.
- U drugom koraku se dodaje bit jedan.
- U trećem koraku se dodaje 64-bitna vrednost dužine poruke.

Spajajući bite pojedinačnih dopuna dobija se ukupna dužina dopune od 224 bita.

4.3. Iterativni heš

Blake 256 je iterativni algoritam i bazira se na opšte prihvaćenoj HAIFA (*Hash Iterative Framework*) strukturi, kao i većina heš funkcija:

$$h_0 = IV$$

$$for\ i=0,..,N-1$$

$$h^{i+1} = compress(h^i, m^i, s^i, t^i)$$

$$return\ h^N$$

gde je N ukupan broj ulaznih blokova. Iterativnost se ogleda u tome da se heš vrednost prethodnog bloka koristi prilikom heširanja sledećeg bloka.

Brojač predstavlja sumu svih bitova informacija koji su bili heširani i koji se trenutno heširaju, ne računajući bitove koji su sastavni delovi dopune. Na primer, ako je originalna (nedopunjena) poruka dužine 600 bita, posle dopune njena dužina postaće 1024 bita zahtevajući dva poziva kompresione funkcije. Prilikom prvog poziva vrednost brojača iznosiće $t^0 = 512$, a prilikom drugog $t^1 = 600$, jer u sastav brojača ne ulaze biti dodati dopunom. Poseban slučaj se javlja kada poslednji blok ne sadrži bite originalne poruke; Takav slučaj je upravo prethodno opisan gde je dužina ulazne poruke 1000 bita. Posle dopune, postojaće 3 bloka ulaznog podatka pri čemu prvi blok poseduje 512 bita korisnih informacija, drugi blok 488 bita korisnih inofrmacija, dok treći sadrži samo bite dopune. Imajući to u vidu vrednost brojača na ulazu kompresione funkcije biće $t^0 = 512$, $t^1 = 1000$ i $t^2 = 0$. Opšte pravilo je: ako poslednji blok ne sadrži bite od originalne poruke, onda se brojač stavlja na nulu.

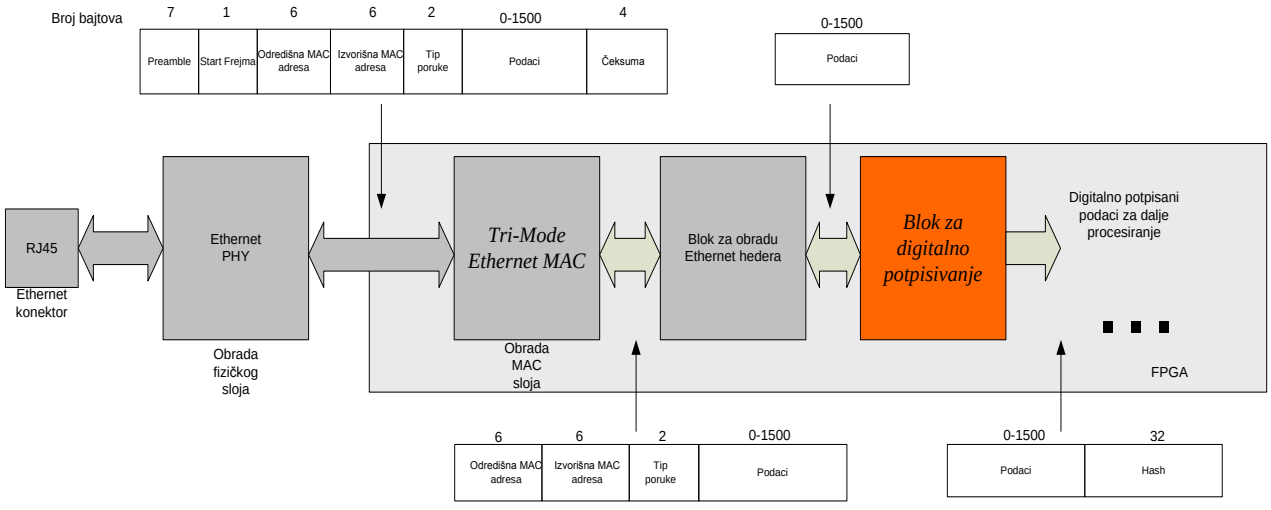
5. STRUKTURA SISTEMA I IMPLEMENTACIJA DIGITALNOG POTPISA IP PAKETA

U ovom poglavlju biće opisana hardverska implementacija bloka za digitalno potpisivanje. Sistem je projektovan korišćenjem VHDL programskog jezika. Osnovne karakteristike sistema su:

- Obrada podataka u striming modu pri čemu između dva susedna IP paketa mora postojati pauza koje će biti definisana u nastavku rada.
- Sistem je projektovan tako da može da računa heš vrednost IP poruke proizvoljne dužine.
- Sistem proverava da li ulazna poruka ima validnu kontrolnu sumu. Ukoliko kontrolna suma nije validna takav IP paket će biti odbačen.
- Sistem neće procesirati poruku ukoliko ne pripada internet protokolu verzija 4.
- Sistem računa heš vrednost koju dodaje na kraj prvobitne poruke. Novodobijena poruka ima ažuriranu vrednost bajtova koji definišu ukupnu dužinu poruke i kontrolnu sumu u IP zaglavlju.
- Sistem može lako da se integriše i koristi u implementaciji 1000/100/10 Mbs Etherneta
- što će biti pokazano u nastavku rada.

5.1. Interfejs bloka za digitalno potpisivanje

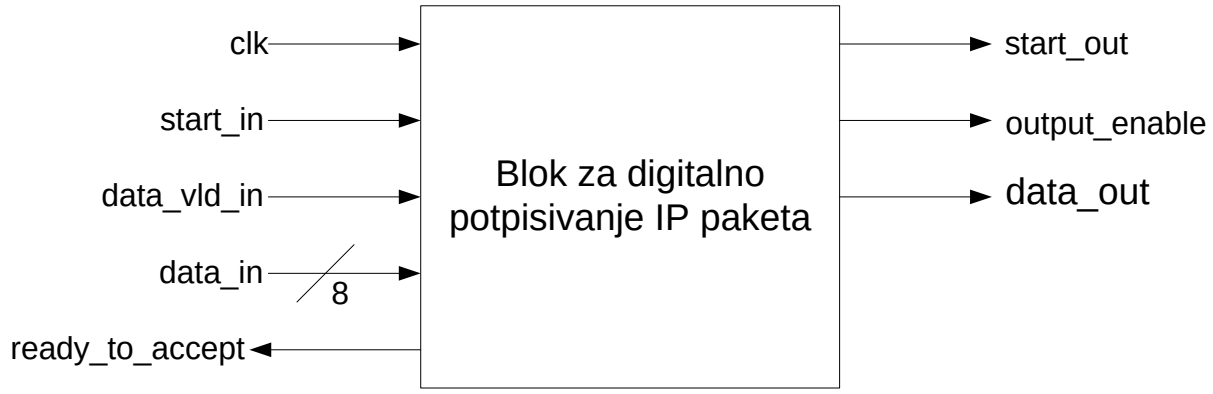
Ulazni interfejs projektovanog sistema je definisan na osnovu izlaznog interfejsa *Tri-Mode Ethernet MAC* jezgra [5]. Osnovni cilj je omogućiti jednostavno povezivanje implementiranog sistema za digitalno potpisivanje IP paketa sa *Ethernet MAC* jezgrom. Između ova dva bloka mora postojati blok koji će obrađivati bajtove sadržane u Ethernet zaglavlju i vršiti filtriranje po tipu poruke. Slika 5.1.1 pokazuje mesto implementiranog bloka u sistemu.



Slika 5.1.1. Povezivanje bloka za digitalno potpisivanje

Iza svakog bloka je pokazano kako izgledaju podaci posle obrade. Uloga nedostajućeg bloka za obradu Ethernet zaglavlja je da odbaci bajtove koji čine odredišnu MAC adresu, izvorišnu MAC adresu i tip poruke. Po standardu maksimalan broj bajtova koji može stići na ulaz bloka za digitalno potpisivanje iznosi 1500 bajtova. Pošto se digitalno potpisivanje radi po Blake 256 algoritmu koji kao krajnji heš daje rezultat od 32 bajta, dužina poruke koja se očekuje na ulazu bloka za digitalno potpisivanje je 1468 bajtova. Na taj način novoformirana poruka ima dužinu 1500 bajtova i može se dalje slati drugim periferijama pomoću Ethernet interfejsa.

Na slici 5.1.2 je prikazan interfejs bloka za digitalno potpisivanje. Sistem nema nezavisno izvučen reset signal zbog činjenice da se svi bitni parametri sistema resetuju na početku svake nove IP poruke koristeći *start_in* signal.



Slika 5.1.2. Interfejs bloka za digitalno potpisivanje

Detaljniji opis signala dat je u tabeli 5.1.1.

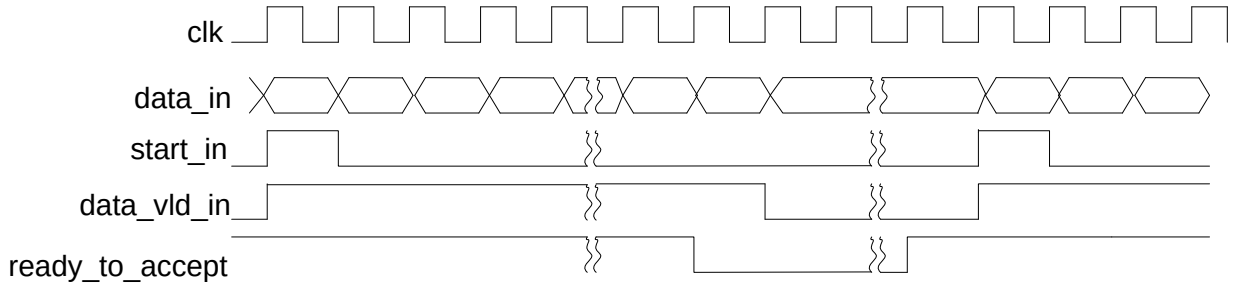
Tabela 5.1.1. Opis ulaznih i izlaznih signala bloka za digitalno potpisivanje

Signal	Direkcija	Opis
clk	Ulaz	Takt iznosi 125 MHz. Ovaj takt u opštem slučaju dolazi iz *Ethernet MAC* jezgra.
data_in [7:0]	Ulaz	Podaci za digitalni potpis
start_in	Ulaz	Kontrolni signal za data_in port. Signalizira transfer prvog podatka.
data_vld_in	Ulaz	Kontrolni signal za data_in port. Signalizira da je podatak validan.
ready_to_accept	Ulaz	*Handshake* signal. Signalizira da je blok u stanju da prihvati podatak sa ulaza.
data_out	Izlaz	Izlazni podatak
start_out	Izlaz	Kontrolni signal za izlazni podatak. Signalizira transfer prvog podatka.
output_enable	Izlaz	Kontrolni signal za izlazni

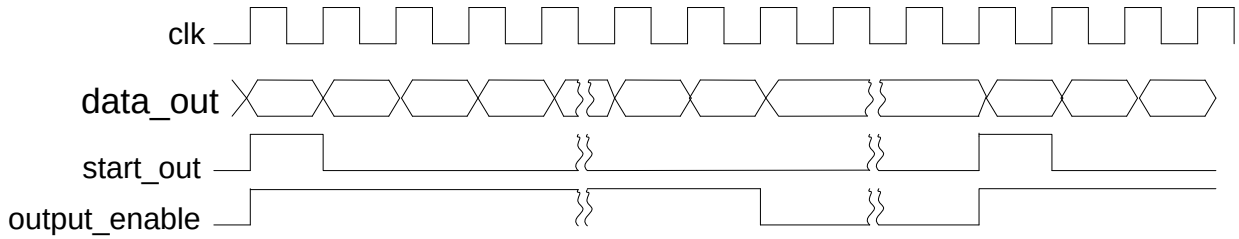
		podatak. Signalizira da je podatak validan.

Vremenski dijagrami signala na ulazu i izlazu prikazani su na slikama 5.1.2 i 5.1.3 respektivno.

Protok podataka na ulazu se kontroliše signalom *ready_to_accept*. Samo u trenutku kada je taj signal aktivan, blok za digitalno potpisivanje je spreman da prihvati podatak sa ulaza. Signal *start_in* određuje granicu između dva paketa, tj kada je on aktivan u toku je prenos prvog bajta IP paketa. Podaci na ulazu se smatraju validnim samo ako je signal *data_vld_in* aktivan (slika 5.1.2.).



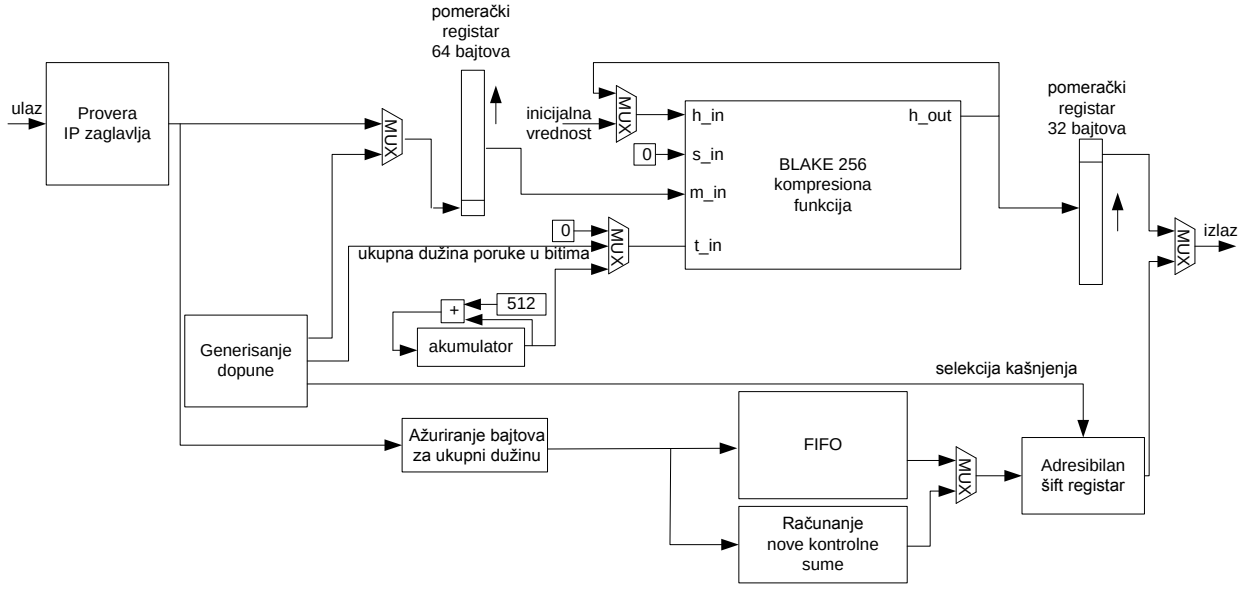
Slika 5.1.2. Vremenski dijagrami signala na ulazu



Slika 5.1.3. Vremenski dijagrami signala na izlazu

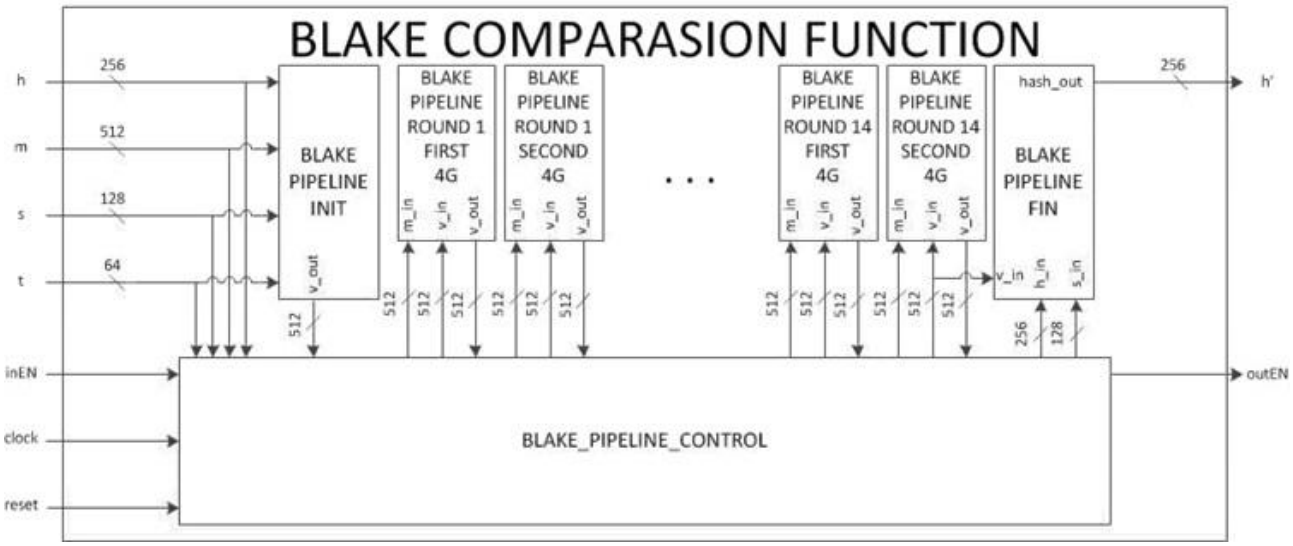
Na osnovu izlaznog interfejsa (slika 5.1.3.) može se primetiti da blok koji se nadovezuje na blok za digitalno potpisivanje IP poruka mora biti u stanju da prihvati podatke u bilo kom trenutku, budući da ne postoji skladistenje podataka u samom bloku za digitalno potpisivanje. Aktivna vrednost *start_out* signala pokazuje trenutak prenosa prvog bajta novoformiranog paketa. Podaci na *data_out* magistrali su validni samo ako je i signal *output_enable* aktivan.

5.2. Implementacija bloka za digitalno procesiranje



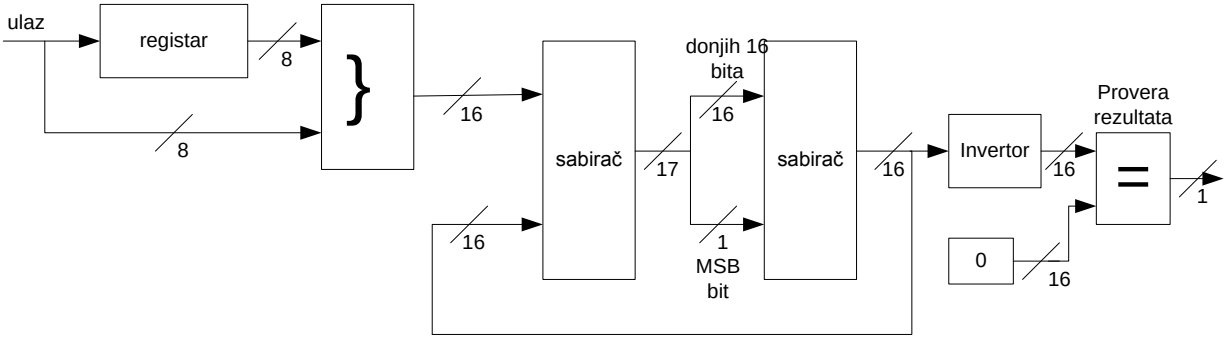
Slika 5.2.1. Blok šema bloka za digitalno potpisivanje

Na slici 5.2.1 je prikazan blok dijagram hardverske implementacije digitalnog potpisivanja korišćenjem BLAKE 256 algoritma. Kod koji implementira Blake 256 kompresionu funkciju je preuzet [3]. Od ponuđene tri implementacije u radu autor se odlučio za pajplajn implementaciju koja omogućava visok stepen paralelizacije, a samim tim i visok protok. Pošto se koristi pajplajn tehnika, kombinaciona logika pojedinih etapa je manja, i samim tim bi trebala da se postigne viša radna frekvencija što doprinosi većem protoku realizacije Blake heširanja. Originalna pajplajn implementacija istovremeno može da procesira 28 paralelnih tokova podataka i prikazana je na slici 5.2.2. Blake algoritam se sastoji iz 14 rundi, pri čemu je svaka runda podeljena na dve podrunde. Svaka podrunda se sastoji od po 4 G funkcije koje mogu da se izvršavaju u paraleli.



Slika 5.2.2. Funkcija kompresije pajplajn implementacije IP BLAKE jezgra

Prilikom implementacije bloka za digitalno potpisivanje autor se susreo sa problemom rutiranja dizajna na 125 MHz. Vremenska analiza u ISE razvojnom okruženju je pokazala da originalna pajplajn implementacija ne zadovoljava željene vremenske kriterijume. Iz tog razloga je povećan broj paralelnih tokova tako što je u implementaciji svake G funkcije dodat registar. Na taj način je ukupan broj paralelnih tokova povećan sa 28 na 56 čime je razvojnom alatu olakšano rutiranje. Naravno ova promena je dovela do povećanja količine resursa koji se koriste na čipu.



Slika 5.2.3. Blok za računanje kontrolne sume

Podaci sa ulaza se najpre dovode na blok za proveru IP zaglavlja. Provera IP zaglavlja se vrši radi provere da li je paket stigao bez grešaka u zaglavlju. Ako postoje greške, paket se odbacuje. Provera podrazumeva proveru vrednosti odgovarajućih polja u okviru zaglavlja i to proveru verzije IP protokola kao i proveru kontrolne sume, čijom proverom se utvrđuje da li postoje bitske greške u zaglavlju. Blok za digitalno potpisivanje paketa je projektovan tako da podržava verziju 4 internet protokola. Iz tog razloga paketi koji ne pripadaju pomenutoj verziji neće se procesirati na izlaz.

Slika 5.2.3 pokazuje blok šemu modula za proveru kontrolne sume nadolazećeg IP datagrama. Podaci sa ulaza se najpre spajaju u 16-bitne reči. Potom se spojena reč sabira sa trenutnom vrednošću zbira. Rezultat sabiranja je 17-bitna reč, pa se donjih 16 bita sabiraju sa nosiocem sume dajući vrednost koja se dovodi povratnom spregom na dalje formiranje ukupnog zbira. Invertovan krajnji zbir se poredi sa nulom gde je u slučaju jednakosti kontrolna suma validna. IP poruka sa nevalidnom kontrolnom sumom neće biti procesirana na izlaz.

Dalja obrada u slučaju ispravnog okvira se može grubo podeliti na:

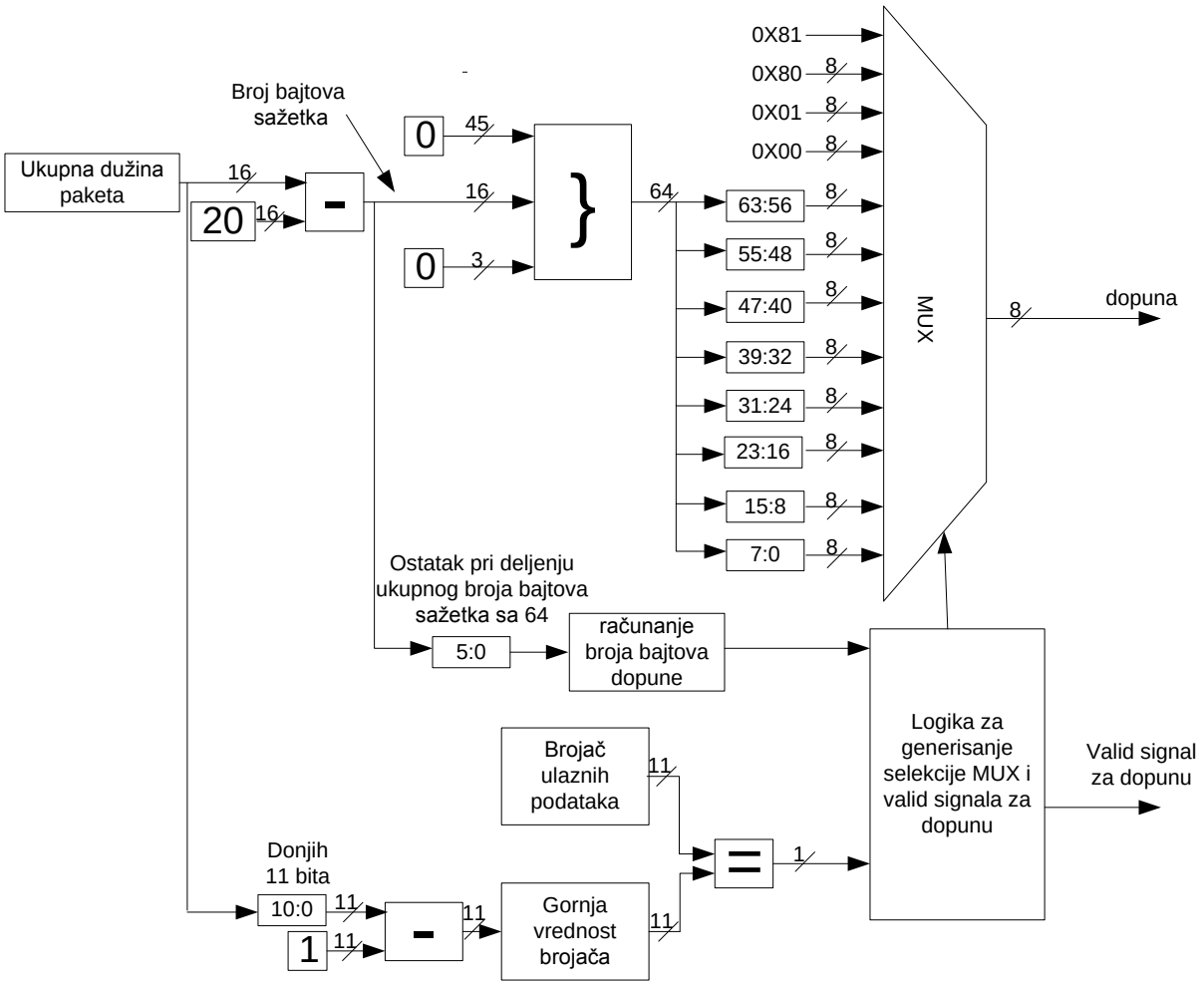
- Deo koji vrši obradu sažetka tj. računanje heš vrednosti sažetka na osnovu Blake 256 algoritma
- Deo koji vrši promenu zaglavlja tj bajtova koji predstavljaju ukupnu dužinu IP poruke i promenu kontrolne sume

Blake 256 kompresiona funkcija zahteva 64-bajtnu ulaznu poruku i zbog toga se bajtovi sa ulaza pakuju u 64-bajtni pomerački registar. U slučaju da broj bajtova sažetka IP okvira nije celobrojni umnožak broja 64, u pomerački registar se dopisuju bajtovi dopune. Selekcija multipleksera na ulazu u pomerački registar definiše da li bajtovi dolaze sa ulaza ili pak iz bloka za generisanje bajtova dopune. Vrednost salta za potrebe ove implementacije je fiksirana na nulu. Ulazna heš vrednost jednaka je inicijalnoj vrednosti specificiranoj po Blake 256 standardu, kada se Blake kompresiona funkcija prvi put poziva u okviru jedne IP poruke, odnosno rezultatu prethodnog poziva kompresione funkcije u ostalim slučajevima. Vrednost brojač promenljive

predstavlja koristan broj bita koji je do tog trenutka bio obrađen. On se prilikom svakog sledećeg poziva uvećava za 512. Tako prilikom prvog poziva kompresione funkcije brojač ima vrednost 512, prilikom drugog poziva 1024 i tako redom. Vrednost brojača u poslednjem pozivu je ukupan broj bita koji je procesiran ne računajući bajtove dopune. Specijalan slučaj se dešava kada je broj bajtova dopune veći od 64 bajta. U tom slučaju vrednost brojača u pretposlednjem pozivu kompresione funkcije jednaka je ukupnom broju bita koji je procesiran ne računajući bajtove dopune, a vrednost brojača u poslednjem pozivu jednaka je nuli, što je i specificirano standardom. Kada se izvrši procesiranje poslednje kompresione funkcije u okviru jedne IP poruke, izlazna vrednost heša se upisuje u 32-bajtni pomerački registar. Najviši bajt pomeračkog registra se dovodi na izlazni multiplekser gde se spaja sa inicijalnom porukom koja je minimalno modifikovana (promenjeni su bajtovi koji definišu dužinu IP poruke i kontrolnu sumu).

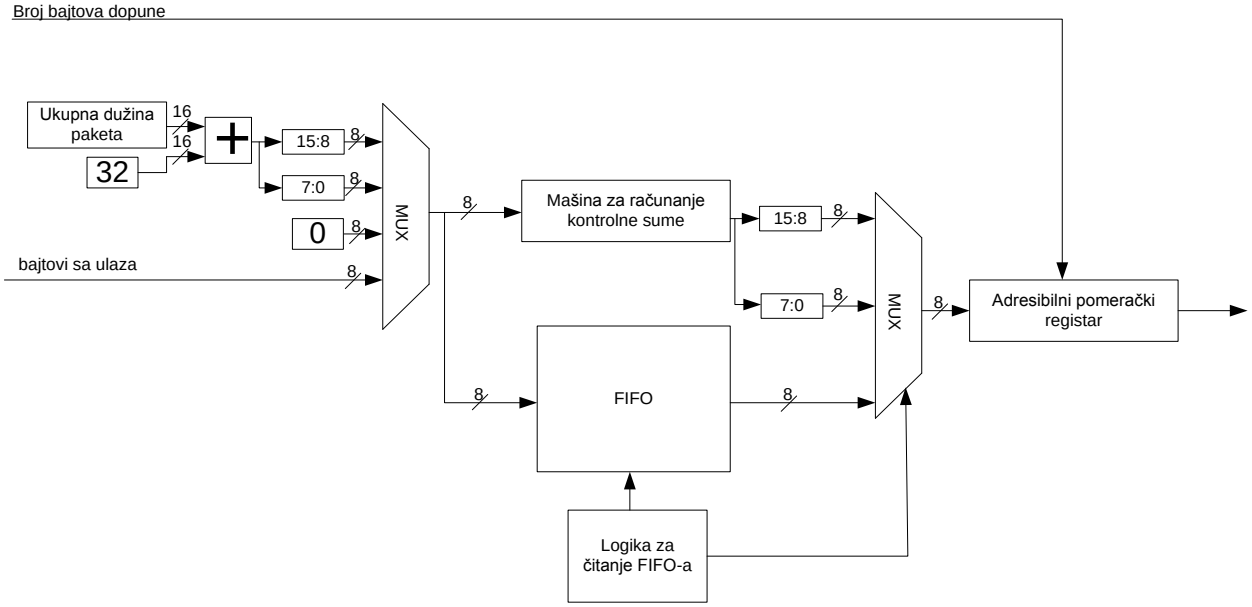
Osnovna uloga bloka za generisanje dopune je da u zavisnosti od dužine IP poruke formira bajtove dopune. Kao što je već u potpoglavlju 4.2 i opisano, na kraj poruke se, najpre, dodaje bit 1 praćen određenim brojem bitova 0, zatim bit 1, i na kraju 64-bitna big-endian reprezentacija dužine poruke. Pošto blok za generisanje digitalnog potpisa vrši procesiranje na bajtovskom nivou, a imajući u vidu minimalnu dužinu dopune od 67 bita definisanu standardom, minimalna dopuna u našem sistemu iznosi 9 bajta što predstavlja 72 bita. Maksimalan broj bajtova dopune jednak je 72 i on se javlja u slučaju da je broj bajtova do dopune prvog punog broja od 64 bajta jednak 8.

Slika 5.2.4 pokazuje blok šemu modula za generisanje dopune. Brojač ulaznih podataka se resetuje na početku svake poruke i broji pristigle bajtove ulazne poruke. Vrednost do koje brojač broji se dobija tako što se dužina poruke definisane u IP zaglavlju umanji za jedan. Kada brojač odbroji do definisane granične vrednosti, svi bajtovi ulazne poruke su primljeni, i može se startovati proces generisanja dopune. Blok koji je na slici obeležen kao logika za generisanje selekcije multipleksera i valid signala za dopunu, realizovana kao brojačka mašina, se u tom trenutku resetuje na nulu.



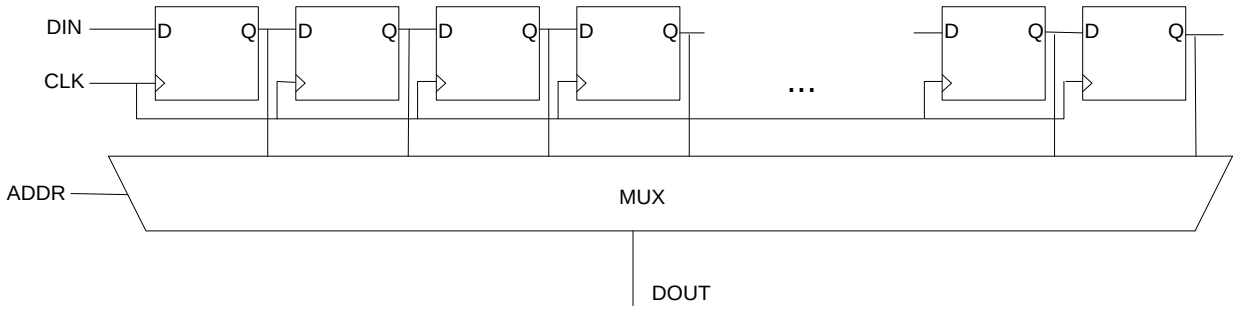
Slika 5.2.4. Blok za generisanje dopune

Brojačka mašina broji do vrednosti koja se dobija kada se ukupan broj bajtova dopune umanji za jedan. Broj bajtova dopune se dobija na osnovu ostatka deljenja broja bajtova sažetka sa 64. Broj bajtova sažetka se računa kao razlika ukupne dužine poruke i broja bajtova zaglavlja. U slučaju da je ostatak veći od 55 ukupan broj bajtova dopune se dobija kao razlika broja 128 i ostatka. U ostalim slučajevima ukupan broj bajtova dopune se dobija oduzimanjem broja 64 i ostatka. Vrednost brojačke mašine definiše selekciju multipleksera tako da je vrednost prvog bajta koji se propušta jednaka 0x80 ili 0x81. Vrednost 0x81 predstavlja specijalan slučaj kada je broj bajtova dopune jednak minimalnom mogućem. Bajt 0x01 se propušta u trenutku kada je vrednost brojačke mašine jednaka vrednosti koja se dobija kao razlika ukupnog broja bajtova dopune i broja 10. Najviši bajt ukupnog broja bita koji se hešira se propušta u trenutku kada je vrednost brojačke mašine jednaka razlici ukupnog broja bajtova dopune i broja 9. Isti postupak se primenjuje i za ostale bajtove ukupnog broja bita poruke. Najniži bajt se procesira na izlaz multipleksera u trenutku kada je vrednost brojačke mašine jednaka vrednosti razlike ukupnog broja bajtova dopune i broja 2.



Slika 5.2.5. Blok za procesiranje zaglavlja

Na slici 5.2.5 je prikazan blok za obradu IP zaglavlja. Mašina za računanje nove kontrolne sume je identična onoj koja je korišćena za proveru kontrolne sume ulaznih bajtova. Njoj se sada kao ulaz dovode ulazni bajtovi pri čemu se na mestu bajtova koji čine ukupnu dužinu poruke dovodi nova vrednost koja predstavlja uvećanu početnu dužinu poruke za 32 bajta. Tih 32 bajta predstavljaju heš vrednost dodatu na kraj nove IP poruke. Na mestu bajtova kontrolne sume se dovode nule. Da bi se izračunala nova kontrolna suma potrebno je da stigne 20 bajtova sa ulaza koji su sastavni delovi IP zaglavlja. Za to vreme je potrebno negde čuvati te podatke dok kontrolna suma ne bude izračunata. U tu svrhu korišćena je FIFO (First In First Out) memorija. U trenutku kada se dvadeseti ulazni bajt upiše u FIFO može se sa sigurnošću znati da su ispunjeni svi uslovi da se kontrolna suma može izračunati pa i otpočinje čitanje zaglavlja. Osnovni zadatak kontrolne logike za čitanje FIFO-a je da startuje čitanje kada se prvih 20 bajtova upišu i da čitanje nastavi sve dok u FIFO-u ima elemenata. Takođe ona generiše signal selekcije multipleksera tako da u trenutku kada bajtovi kontrolne sume izlaze iz FIFO-a, multiplekser prosledi dalje bajtove nove kontrolne sume. Adresibilan pomerački registar ima zadatak da zakasni podatke tako da u trenutku kada se poslednji bajt ulazne poruke pojavi na ulazu izlaznog multipleksera sa slike 5.2.1, heš vrednost već bude upisana u pomerački registar i bude spremna za slanje. Vrednost kašnjenja je promenljiva i isključivo zavisi od broja bajtova dopune. Ukoliko je broj bajtova sažetka takav da dopuna nije potrebna (broj bajtova sažetka je deljiv sa 64) kašnjenje adresibilnog pomeračkog registra je 38. Ovu vrednost autor je dobio na osnovu vremenskih dijagrama generisanih simulacijom. Broj bajtova dopune definiše koliko je potrebno povećati kašnjenje. Za svaki dodatni bajt dopune kašnjenje se povećava za jedan takt. Tako na primer ukoliko je broj bajta dopune 10, kašnjenje ovog bloka treba da bude 48. Arhitektura adresibilnog pomeračkog registra prikazana je na slici 5.2.6.



Slika 5.2.6. Blok šema adresibilnog pomeračkog registra

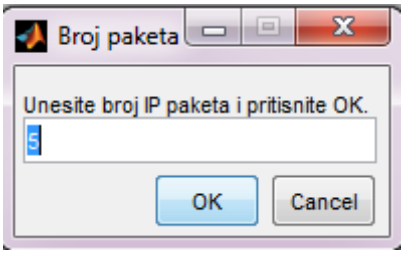
Autor je na osnovu vremenskih dijagrama simulacije zaključio da različito kašnjenje adresibilnog pomeračkog registra za dve sukcesivne IP poruke, jeste razlog zbog kojeg mora postojati razmak između poruka veći od teorijskog. Teorijski minimalan razmak između dve IP poruke iznosi 32 takta, jer je izlazna poruka za 32 bajta veća. Ukoliko bi sistem bio projektovan za fiksnu dužinu IP poruke blok za digitalno potpisivanje bi mogao podržati minimalan teorijski razmak između dva paketa od 32 takta. Kako u opštem slučaju to nije tako, putem simulacije, puštanjem velikog broja paketa jednog za drugim, sa različitim brojem bajtova dopune, ustanovljeno je da za razmak od 108 taktova između dve IP poruke sistem funkcioniše ispravno.

6. VERIFIKACIJA DIZAJNA I ANALIZA PERFORMANSI

Ovo poglavlje opisuje način verifikacije bloka za digitalno potpisivanje IP paketa. Takođe je dat pregled performansi za urađenu FPGA implementaciju.

6.1. Matlab referentna simulacija

Uloga MATLAB referentne simulacije je da simulira ponašanje bloka za digitalno potpisivanje. Referentna simulacija na početku generiše IP poruke i čuva ih u fajlu *input.txt*. Ukupan broj poruka koji će biti generisan definiše korisnik u prozoru koji se javlja prilikom pokretanja koda kao što je prikazano na slici 6.1.1. Dužina pojedinačnih IP poruka, kao i sadržaj sažetka se generiše korišćenjem MATLAB-ove *rand* funkcije.



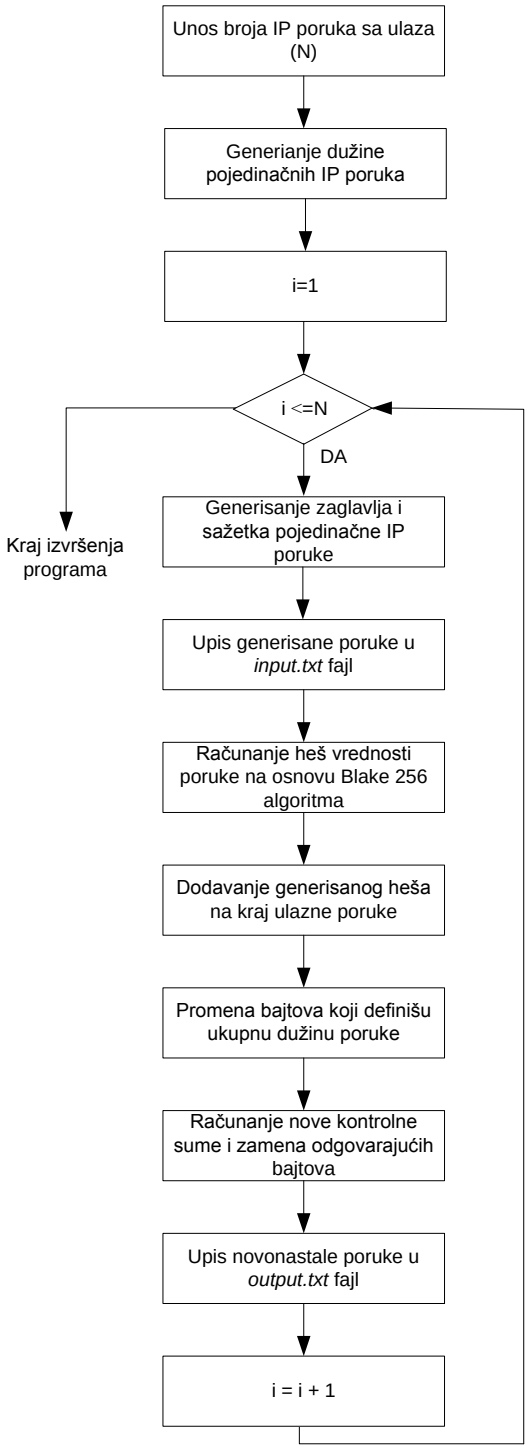
Slika 6.1.1. Korisnički prozor

Bajtovima IP zaglavlja, su za potrebe simulacije, dodeljene vrednosti shodno tabeli 3.1.1.

Tabela 3.1.1. IP zaglavlje

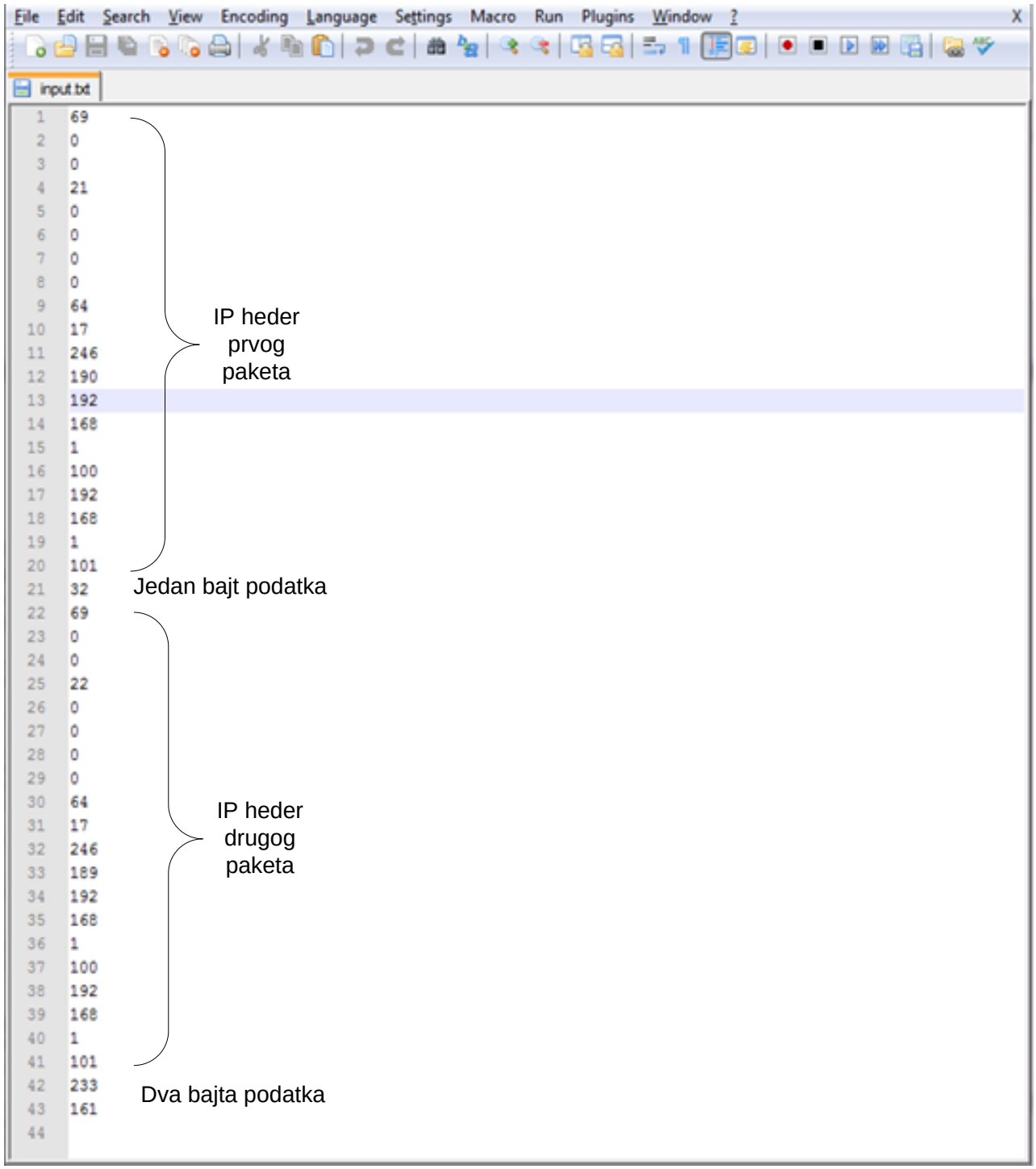
Bajt	Funkcija	Vrednost
1.	Verzija i dužina zaglavlja	0x45
2.	Tip servisa	0x00
3. i 4.	Ukupna dužina poruke uključujući i zaglavlje	Promenljiva i zavisi od dužine generisane poruke
5. i 6.	Identifikacija	0x00 i 0x00
7. i 8.	Flegovi i fragmentacija	0x00 i 0x00
9.	Vreme života	0x40
10.	Protokol	0x11 (UDP)
11. i 12.	Kontrolna suma	Vrednost zavisi od ostalih bajtova IP zaglavlja
13. , 14. , 15. i 16.	Izvorišna IP adresa	0xC0, 0xA8, 0x01 i 0x64 (192.168.1.100)
17. , 18. , 19. i 20.	Odredišna IP adresa	0xC0, 0xA8, 0x01 i 0x65 (192.168.1.101)

Generisane IP poruke se potom propuštaju kroz kod koji vrši računanje heša na osnovu Blake 256 algoritma. Izračunati heš svake IP poruke se dodaje na njen kraj i vrši promena bajtova u zaglavlju koji definišu ukupnu dužinu poruke i kontrolnu sumu. Novonastale IP poruke se potom čuvaju u *output.txt* fajlu. Algoritam izvršavanja MATLAB koda je prikazan na slici 6.1.2. Deo koda koji se tiče računanja heš vrednosti ulazne poruke na osnovu Blake 256 algoritma autor je preuzeo sa sajta https://131002.net/blake/ [6].



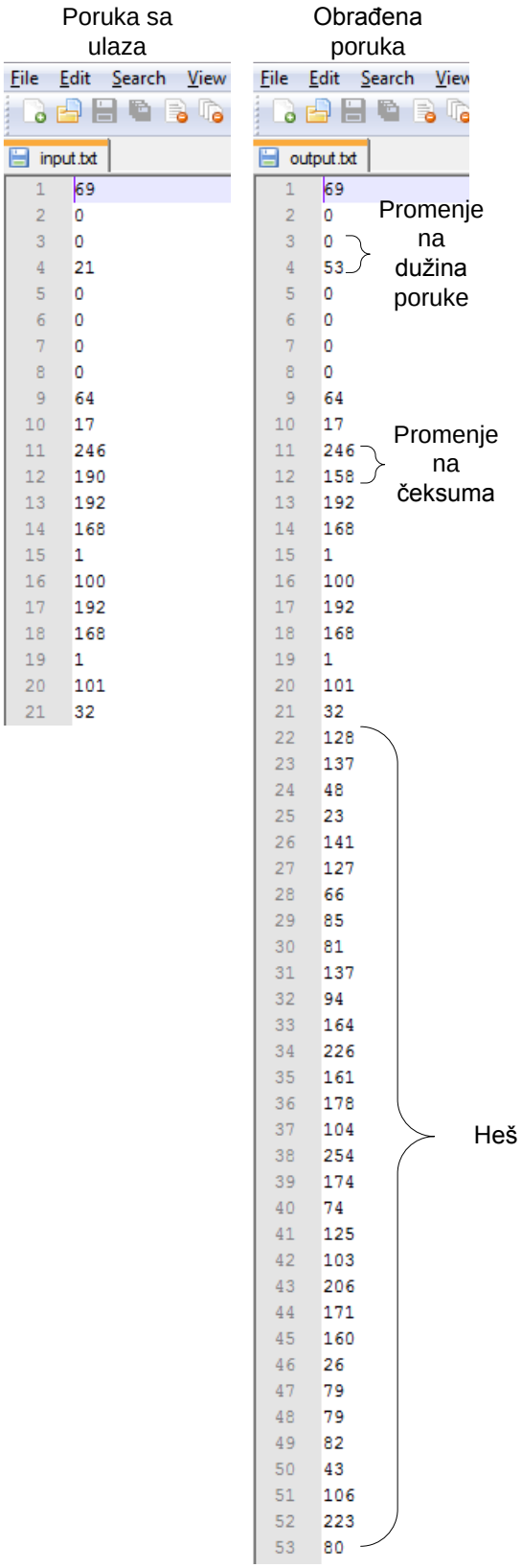
Slika 6.1.2. Algoritam izvršavanja MATLAB referentne simulacije

Na slici 6.1.3 je prikazan izgled *input.txt* fajla u kome su sacuvana 2 IP paketa dužine sažetka jedan i dva bajta respektivno.



Slika 6.1.3. Izgled *input.txt* fajla

Izgled prve IP poruke iz fajla *input.txt*, kao i obrađene poruke iz fajla *output.txt* je prikazan na slici 6.1.4.



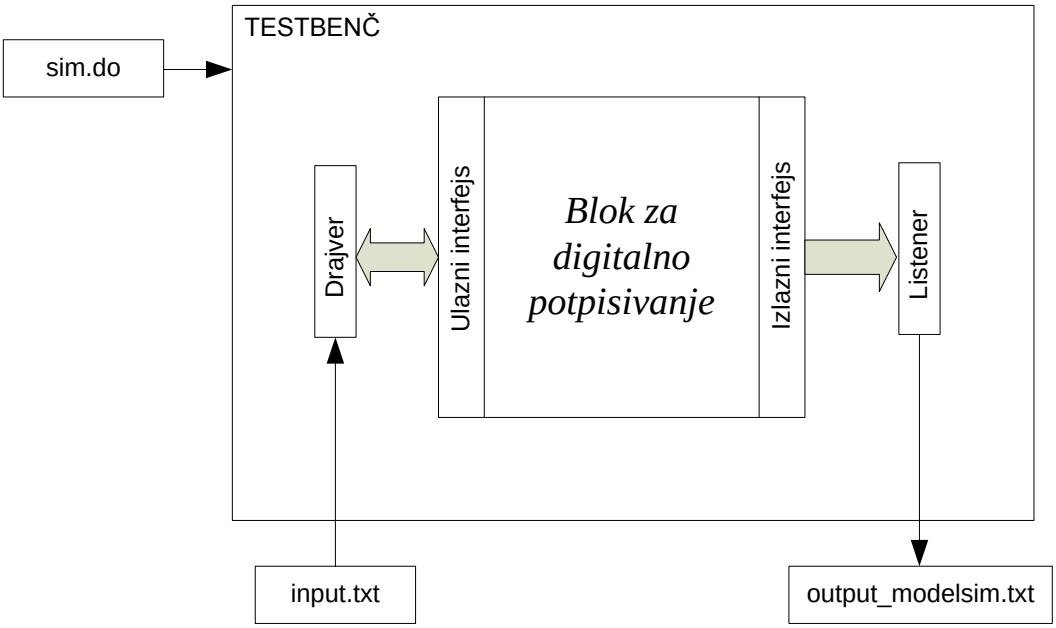
Slika 6.1.4. Poređenje *input.txt* i *output.txt* fajla

6.2. Testbenč

Testbenč predstavlja logiku, napisanu u VHDL jeziku za opis hardvera, koja okružuje dizajn koji se testira i koja vrši određene pobude na ulazu u dizajn i prati da li izlazi imaju predviđeno ponašanje.

Za simulaciju se koristio Modelsim 10c, dok je za generisanje ulaznih i očekivanih izlaznih podataka korišćena već unapred opisana MATLAB referentna simulacija.

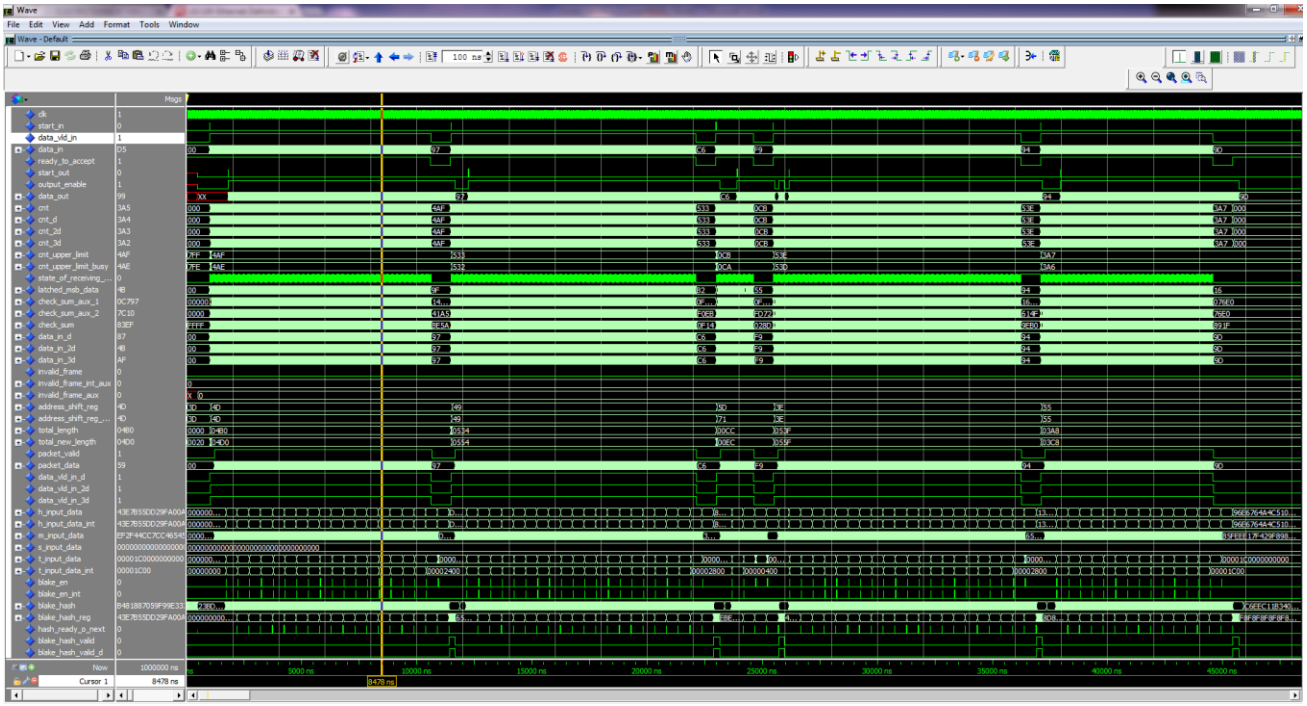
Donja slika pokazuje testbenč arhitekturu. Pokazuje komponente test okruženja kao i interfejse prema bloku za digitalno potpisivanje IP paketa. Fajl koji drajver koristi kao ulazni je *input.txt* generisan MATLAB referentnom simulacijom. *Listener* blok prihvata podatke sa izlaza i čuva ih u fajlu *output_modelsim.txt*. Po završetku simulacije fajl *output.txt* dobijen MATLAB referentnom simulacijom i fajl *output_modelsim.txt* generisan u testbenču se porede korišćenjem alata za poređenje teksta kao što je na primer *Beyond Compare*. Identičnost ta dva fajla nam garantuje ispravnu funkcionalnost modula.



Slika 6.2.1. Arhitektura test okruženja

Kompajliranje VHDL fajlova i prikazivanje vremenskih dijagrama u Modelsimu se postiže pokretanjem *sim.do* skripte u samom Modelsim okruženju. Sama skripta se sastoji iz komandi koje na početku vrše kompajliranje svih VHDL fajlova, zatim ubacuju signale koje hoćemo da pratimo u vremenski dijagram signala i na kraju pokreće izvršenje simulacije.

Na slici 6.2.2 je prikazan izgled prozora Modelsim simulatora nakon pokretanja simulacije.



6.2.2. Izgled prozora Modelsim simulatora

6.3. Analiza performansi

6.3.1. Performanse sistema

Projektovan sistem omogućava obradu poruka proizvoljne dužine ali istovremeno zahteva vremensku pauzu između dve poruke od 108 taktova. Ta pauza ograničava maskimalan protok sistema na ulazu. Imajući u vidu maksimalnu specificiranu dužinu poruke zajedno sa zaglavljem od 1468 bajtova maksimalan protok na ulazu koji se može ostvariti jednak je:

$$\frac{1468}{1468 + 108} * 1000Mbs = 931Mbs$$

Ukoliko je akcenat na sistemu da ima maksimalan teorijski moguć protok, on se može jednostavno promeniti da podržava samo maksimalnu dužinu poruke od 1468 bajtova. U tom slučaju rastojanje između dva sukcesivna paketa može iznositi 32 takta. Maksimalan teorijski protok u tom slučaju iznosi:

$$\frac{1468}{1468 + 32} * 1000Mbs = 978.66Mbs$$

6.3.2. Rezultati sinteze

U tabeli 6.3.1 je prikazan rezultat dobijen iz fajlova koje je generisao Xilinx ISE alat posle procesa implementacije za tri vrste čipa. U sva tri slučaja dizajn je uspešno izrutiran bez grešaka u vremenskoj analizi.

Tabela 6.3.1 Rezultati sinteze

Čip	Slajs Registri	Slajs LUT	Ukupno iskoriscenje slajseva
Kintex 7 xc7k325t-2ffg900	48055/407600 11%	32966/203800 16%	10071/50950 19%
Virtex 7 xc7vx330t-2ffg1157	48055/408000 11%	33631/204000 16%	9804/51000 19%
Virtex 6 xc6vlx130t-2ff484	48055/160000 30%	32913/80000 41%	10139/20000 50%

Da bi ISE alat znao na kom taktu treba da izvrši vremensku analizu dizajna napravljen je prost ucf fajl koji definiše radnu frekvenciju takta. Sadržaj ucf fajla dat je u nastavku:

```
NET "clk" TNM_NET = TNM_125MHz;
NET "clk" CLOCK_DEDICATED_ROUTE = FALSE;
TIMESPEC "TS_125MHz" = PERIOD "TNM_125MHz" 8 ns HIGH 50 %;
```

7. ZAKLJUČAK

Ovaj rad predstavlja realizaciju projekta digitalnog potpisa na bazi Blake-256 algoritma. Projektovani kod omogućava digitalno potpisivanje poruka koje se šalju brzinama do 1Gbit/s. Ulazni podaci (kao i izlazni) su organizovani u vidu bajtova, što značajno pojednostavljuje kod i organizaciju podataka u memoriji FPGA kola. Algoritam potpisivanja je realizovan na bazi pajplajn arhitekture čime se minimizuje kašnjenje paketa kroz FPGA kolo (na račun povećanja zauzetosti resursa u njemu).

Inkorporirani kod omogućava takvu organizaciju hardvera (rutiranje FPGA kola) da se u njemu paralelno realizuju:

- provera ispravnosti zaglavlja pristiglog paketa (na osnovu identifikacije verzije protokola, dužine zaglavlja i kontrolne sume dvobajtnih reči u zaglavlju).
- separaciju zaglavlja od korisnog dela poruke.
- promenu dvobajtne reči koje predstavljaju dužinu poruke (smeštene između 16-tog i 31-og bita zaglavlja). Ova promena se vrši na osnovu produžetka poruke za 32 bajta (u kojima je smeštena vrednost heša).
- promenu dvobajte reči koja predstavljaju kontrolnu sumu zaglavlja (zbog izmene vrednosti dužine poruke).
- dopunjavanje korisnog dela poruke do umnoška od 64-bajta (*padding*), neophodnog za primenu Blake algoritma.
- generisanje finalnog heša (kao i heša u sukcesivnim koracima kompresije poruke).
- sinhronizaciju slanja novoformiranog zaglavlja sa korisnim podacima poruke (dopunjenim sa 32 bajta heša).

Ovako programiran hardver za digitalno potpisivanje ne podrazumeva šifrovanje heša privatnim i javnim ključevima, ali se ta operacija može jednostavno obaviti nad poslednja 32 bajta poruke. Neki od ovih algoritama (kao i sam Blake 256 algoritam) već su dati u nekim od prethodnih master radova [3].

Funkcionalna i vremenska simulacija potvrđuju da je vreme dolaska 64 bajta poruke dovoljno dugačko da pajplajn Blejk-256 algoritam može da generiše heš vrednost iz prethodnih 64 bajta pre nego što stigne sledeći segment od 64 bajta (512 bita). To omogućava kontinualnu obradu paketa u realnom vremenu bez potrebe za velikom memorijom FPGA kola. Kao rezultat takve obrade broj poruka koje se mogu potpisivati nije ograničen u vremenu. Jedino ograničenje koje postoji je da između dve poruke postoji razmak od stotinak taktova i on zavisi od konkretnog sadržaja poruke i pretpostavka je da bi se dodatnim resursima i ovaj nedostatak mogao kompenzovati.

Primer potrošnje resursa na različitim FPGA kolima familije Xilinx pokazuje da su resursi koje okupira programirani hardver prihvatljivi čak i kada je struktura za digitalno potpisivanje realizovana u pajplajn konfiguraciji. Pored toga pretpostavka je da se obrada vršila na taktu od

125MHz. Kako savremena FPGA kola rade sa taktovima od preko 600 MHz realna pretpostavka je da bi se (uz izvesne modifikacije) programirana arhitektura mogla primeniti i na 10 Gbit-ne pakete.

LITERATURA

[1] https://en.wikipedia.org/wiki/IPv4_header_checksum (26.03.2016.)

[2] Olakunle Esuruoso, *High Speed FPGA Implementation of Cryptographic Hash Function* http://scholar.uwindsor.ca/cgi/viewcontent.cgi?article=1123&context=etd (10.04.2016.)

[3] Goran Ognjanović, *Hardverska implementacija BLAKE algoritma za heširanje* http://telekomunikacije.etf.rs/predmeti/te4ks/master.php (10.04.2016.)

[4] Aumasson, J.P. , Henzen L. , Meier W. , Phan R.C.W, *SHA-3 proposal BLAKE* https://131002.net/blake/blake.pdf (23.03.2016.)

[5] Vivado Design Suite, *Tri-Mode Ethernet MAC v8.3* http://www.xilinx.com/support/documentation/ip_documentation/tri_mode_ethernet_mac/v8_3/pg051-tri-mode-eth-mac.pdf (26.03.2016.)

[6] Thomas Burgess, Joseph Jelley, David Smith, Claire Weston, *MATLAB implementation of BLAKE-256, non-object-oriented* https://131002.net/blake/BLAKE256_matlab.zip (10.04.2016.)