**VISOKA ŠKOLA ZA INFORMACIJSKE TEHNOLOGIJE**

**STRUČNI STUDIJ INFORMACIJSKE TEHNOLOGIJE**

**ZAGREB**

**DIPLOMSKI RAD**

**USPOREDBA POSTUPAKA HASHIRANJA (SHA1, SHA256, MD5) I NJIHOVA PODRŠKA U .NET FRAMEWORKU**

**Petar Domitrović**

**Zagreb, mjesec 2019. godine**

Sadržaj

[POPIS SLIKA 3](#_Toc177656278)

[POPIS TABLICA 5](#_Toc177656279)

[1. UVOD 6](#_Toc177656280)

[2. POSTUPCI HASHIRANJA 8](#_Toc177656281)

[2.1 HASHIRANJE 8](#_Toc177656282)

[2.2 MD5 8](#_Toc177656283)

[2.2.1 Povijest 8](#_Toc177656284)

[2.2.2 Upotreba 8](#_Toc177656285)

[2.2.3 Algoritam 9](#_Toc177656286)

[2.3 SHA-1 10](#_Toc177656287)

[2.3.1 Povijest 10](#_Toc177656288)

[2.3.2 Upotreba 10](#_Toc177656289)

[2.3.3 Algoritam 10](#_Toc177656290)

[2.4 SHA-256 11](#_Toc177656291)

[2.4.1 Povijest 11](#_Toc177656292)

[2.4.2 Upotreba 11](#_Toc177656293)

[2.4.3 Algoritam 12](#_Toc177656294)

[2.5 Sličnosti i razlike 13](#_Toc177656295)

[3. Implementacija u .NET Frameworku 15](#_Toc177656296)

[3.1 Dostupne implementacije unutar .NET Frameworka 15](#_Toc177656297)

[3.2 Aplikacija 15](#_Toc177656298)

[3.2.1 Glavni izbornik 15](#_Toc177656299)

[3.2.2 Hasher 16](#_Toc177656300)

[3.2.4 Brzina sažimanja 19](#_Toc177656301)

[4. PRAKTIČNI RAD-PRIKAZ U SLIKAMA I TABLICAMA DIJELOVA IZRADE ZAVRŠNOG I DIPLOMSKOG RADA 22](#_Toc177656302)

[4.1 Frontend 22](#_Toc177656303)

[4.1.2 Prozor „Hasher 22](#_Toc177656304)

[4.1.3 Prozor „Test Brzine“ 22](#_Toc177656305)

[4.2 Backend 24](#_Toc177656306)

[5. ZAKLJUČAK 34](#_Toc177656307)

[LITERATURA 36](#_Toc177656308)

[SAŽETAK 37](#_Toc177656309)

[SUMMARY 38](#_Toc177656310)

# POPIS SLIKA

[Slika 1: Shema za MD5 8](file:///C:\Users\Petar%20Domitrović\OneDrive\Documents\Završni\Rad\Usporedba%20postupaka%20hashiranja%20(SHA1,%20SHA256,%20MD5)%20i%20njihova%20podrška%20u%20.NET%20Frameworku(verzija%2005).docx#_Toc151738678)

[Slika 2: Shema za SHA1 10](#_Toc151738679)

[Slika 3: Shema za SHA256 12](#_Toc151738680)

[Slika 4: Glavni izbornik 15](#_Toc151738681)

[Slika 5: Prozor Hasher 16](#_Toc151738682)

[Slika 6: Rezultat kada je unos prazan odnosno „“ 17](#_Toc151738683)

[Slika 7: Rezultat kada odaberemo datoteku 17](#_Toc151738684)

[Slika 8: Rezultat kada je unos „a“ 18](#_Toc151738685)

[Slika 9: Razlika između rezultata unosa „a“ ( prva 3) i unosa „ab“ (zadnja 3) 18](#_Toc151738686)

[Slika 10: Prozor Test brzine 19](#_Toc151738687)

[Slika 11: Usporedba brzine 3 hasha u obradi lozinke od 8 znakova 20](#_Toc151738688)

[Slika 12: Usporedba brzine 3 hasha u obradi 1GB podataka 20](#_Toc151738689)

[Slika 13: Usporedba brzine svih default hasha u obradi 1GB podataka 21](#_Toc151738690)

[Slika 14: Pomoćne metode za pretvorbu iz byte array u string i obrnuto 22](#_Toc151738691)

[Slika 15: Metoda Hasher() 23](#_Toc151738692)

[Slika 16: Klik na gumb „Pregled“ 23](#_Toc151738693)

[Slika 17: Metoda PickedHash() vraća listu odabranih hasheva 24](#_Toc151738694)

[Slika 18: Metoda HashNames() vraća listu odabranih hasheva 24](#_Toc151738695)

[Slika 19: Metoda MyMD5() 25](#_Toc151738696)

[Slika 20: Metoda RoundsMD5() prvi dio 25](#_Toc151738697)

[Slika 21: Metoda RoundsMD5() drugi dio 26](#_Toc151738698)

[Slika 22: Metoda RoundsMD5() treći dio 26](#_Toc151738699)

[Slika 23: Metoda RoundsMD5() četvrti dio 26](#_Toc151738700)

[Slika 24: Metoda MySHA1() 27](#_Toc151738701)

[Slika 25: Metoda RoundsSHA1() prvi dio 27](#_Toc151738702)

[Slika 26: Metoda RoundsSHA1() drugi dio 28](#_Toc151738703)

[Slika 27: Metoda RoundsSHA1() treći dio 28](#_Toc151738704)

[Slika 28: Metoda RoundsSHA1() četvrti dio 28](#_Toc151738705)

[Slika 29: Metoda MySHA256() 29](#_Toc151738706)

[Slika 30: Metoda RoundsSHA256() prvi dio 29](#_Toc151738707)

[Slika 31: Metoda RoundsSHA256() drugi dio 30](#_Toc151738708)

[Slika 32: Metoda RoundsSHA256() treći dio 30](#_Toc151738709)

[Slika 33: Metoda AppendPadd() 31](#_Toc151738710)

[Slika 34: Metoda ExtractBlock() 31](#_Toc151738711)

[Slika 35: Metoda PrimeNumbers() 32](#_Toc151738712)

[Slika 36: Metoda ConvertByteListToUintList() 32](#_Toc151738713)

[Slika 37: Metoda SwitchEndianness() 33](#_Toc151738714)

[Slika 38: Metoda MessageDigest() 33](#_Toc151738715)

# POPIS TABLICA

[Tablica 1: Usporedba metoda MD5, SHA1 i SHA256 13](#_Toc151738805)

# UVOD

U ovom rade ćete saznati nešto više o hashevima odnosno sažecima. Svrha sažetaka je pretvorba bilo koje količine podataka u fiksni, naočigled nasumični skup znakova. Vrste podataka mogu biti od teksta, lozinki, dokumenata, certifikata i sve do baza podataka. Na kraju pretvorbe odnosno sažimanja dobivamo ključ fiksne veličine npr. 64 znakova u heksadekadskom prikazu. Taj ključ se koristi kao potvrda valjanosti pretvorenih podataka. Ako se dogodi bilo kakva promjena u podatku i ponovno obavimo sažimanje, ključ će biti potpuno drugačiji. Ta osjetljivost daje vrijednost sažetku.

Moguće je izmijeniti podatke tako da daju isti ključ, ali to ovisi o kvaliteti sažetka i zahtjeva puno truda i resursa. Postoje metode kojima se na silu mogu probiti sažetci, ali nisu uvijek isplative. Nesigurni sažetci se izbacuju iz upotrebe i stvaraju se novi koji ih mijenjaju. Tako danas imamo sažetke koji iako nesigurni i dalje služe nekoj svrsi pošto su tehnički ne zahtjevni i osnovna funkcija stvaranja ključa je dovoljna.

Sažetci su rašireniji nego što se čini. Koriste se u raznim internetskim protokolima, pohrani lozinki, održavanju bazi podataka, certifikatima, potpisima, provjeri verzija i u samom programiranju.

U ovom radu su zastupljeni MD5, SHA-1 i SHA-256. Svaki od njih ima prednosti i mane, no na kraju su mane najbitnije, pogotovo kada se radi o sigurnosti.

MD5 je najbrži i zahtjeva najmanje mjesta za pohranu ključa. Svaki današnji laptop neće imati problema probiti sadržaj MD5 sažetka tako da nije dobar za ništa drugo osim provjere podataka.

SHA-1 je nasljednik MD5. Izbačen je iz službene upotrebe isto kao MD5. Kao srednje dijete SHA-1 gubi svrhu i koristi se jedino radi većeg kapaciteta od MD5, odnosno manje šanse za ponavljanje ključa kod obrade veće količine podataka.

SHA-256 je nasljednik SHA-1 i još uvijek je u punoj radnoj snazi. Koristi se u sve svrhe koje se sažetak može koristiti. Postoji nova obitelj sažetaka SHA-3, ali nije namijenjen da direktno zamijeni SHA-256 odnosno SHA-2.

Struktura algoritma svih 3 predstavljenih sažetaka prati istu ideju. Ulazni podaci se nadopunjuju po potrebi i definiraju se konstante. Dalje dijelimo nadopunjene podatke na blokove koji se svaki zasebno obrađuje. Blokovi se obrađuju kroz više rundi radi bolje obrade. Ključ ima početnu vrijednost i dijeli se u riječi koje se mijenjaju kroz runde. Unutar jedne runde postoje različite funkcije i konstante koje mijenjaju riječi. Nakon što je obrađen zadnji blok, riječi se spajaju u ključ koji tražimo.

Vidljiv je napredak u istraživanju između sažetaka po strukturi algoritma i kvaliteti funkcija. Neke stvari koje je SHA-1 zanemario, a MD5 imao se ponovno pojavljuju u SHA-256. SHA-256 također ima dijelove SHA-1 uz svoje originalne dodatke. Sve vrline i mane MD5 i SHA-1 su uzete u obzir pri stvaranju SHA-256.

Ova 3 sažetka je moguće implementirati na mnogo programskih jezika. Odabrana implementacija je u C# odnosno .NET Framework-u. Aplikacija je u formatu Windows programa.

Svrha rada je prikaz sažimanja i ključa koji se dobije izvedbom sažetaka MD5, SHA-1 i SHA-256. U aplikaciji su ubačene postojeće implementacije .NET Framework-a te su dodane originalne. Moguće je izvesti sažimanje nad unesenim tekstom ili odabrati datoteku. Radi jednostavnosti odabir je ograničen na .txt datoteke. Rezultat se može dobiti pojedinačno ili odjednom za svaki sažetak ovisno o želji korisnika. Druga funkcija aplikacije omogućava usporedbu brzine izvedbe različitih sažetaka nad .txt datotekom. Rezultat je prikazan u grafu koji mjeri rezultat u milisekundama. Omogućen je slobodan odabir tako da je moguće mjeriti brzinu jednog sažetka ili usporediti više njih odjednom.

# POSTUPCI HASHIRANJA

## 2.1 HASHIRANJE

Sažimanje (hashiranje) je postupak kojim se na bilo koji način transformira ulazni podatak, obično u obliku niza. Izlazna vrijednost se naziva sažetak (hash) i ona je obično fiksne duljine neovisno o duljini ulazne vrijednosti. Sažetak je izražen u relativno kratkom obliku tako da se lako pohranjuje i koristi. Transformacije u procesu sažimanja mogu biti vrlo jednostavne, ali u profesionalnom okruženju se jako zakompliciraju radi sigurnosti. Slab sažetak se može relativno lako razbiti i doći do originalne vrijednosti prije transformacije. Ni profesionalni sažetak algoritmi rađeni na budžetu SAD-a nisu otporni na probijanje. Samo je potrebno vrijeme za istraživanje i pasivni napredak hardvera. Stoga su novi sažeci neprestano u razvoju.

## 2.2 MD5

## 2.2.1 Povijest

MD5 je 1991. u SAD-u Ronald Rivest dizajnirao kao zamjenu za MD4, a specificiran je1992. Već od 1996. nakon što su vidljive prve slabosti MD5 algoritma kriptografi prelaze na SHA-1. 2005. su demonstrirani napadi koji unutar par sati na običnom laptopu naprave sudar. Sudarom se smatra kada se nađu dvije ulazne vrijednosti koje proizvedu isti sažetak. 2006. je to svedeno na manje od minute. Od 2010. MD5 se smatra enkripcijski slomljenim i neprikladnim za upotrebu.

## 2.2.2 Upotreba

MD5 se danas najčešće koristi za provjeru integriteta podataka. Iz razloga što nije tehnički zahtjevan u odnosu na novije algoritme naime SHA obitelj algoritama. U šifriranju se više ne koristi službeno jer nije siguran. Opet zbog jednostavnosti manje tvrtke i web stranice ga još uvijek koriste za pohranu lozinki.

## 2.2.3 Algoritam

Diagram

Description automatically generated

Slika : Shema za MD5

Slika 1 prikazuje shemu rada za MD5. MD5 radi sa 512 bita. Ovisno o veličini originalne poruke nadopunjuje se s najmanje 1 bitom sve do 512 bitova, Na kraju poruke se uvijek dodaje bit 1. Nakon doga se dodaju 0 dok poruka nije za 64 bita manja od modula 512. U ta 64 bita se upisuje veličina originalne poruke u bitovima. Sve vrijednosti su mali endian. MD5 ima ključ od 128 bita koje dijeli u 4 riječi veličine 32 bita. To su A, B, C i D. Inicijalizirane su na određenu početnu vrijednost. Iz poruke se uzimaju blokovi od 512 bita i počinje obrada. Obrada se izvršava u 64 runde za svaki blok. U 64 runde razlikujemo 4 grupe po 16 rundi u kojima se mijenjaju formule F, G, H , I te konstante Ki. Kroz 64 runde se također mijenjaju konstanta s za lijevu rotaciju (<<<) bitova i 32 bitna riječ Mi na koje dijelimo 512 bitni blok. Sva se zbrajanja () odvijaju po modulu 2^32 .

Formule:

runda 1-16

runda 17-32

runda 33-48

runda 49-64

Nakon formule riječi A,B,C i D se spremaju jedno u drugu kao prikazano na shemi. Poslije svih 64 rundi vrijednosti A,B,C i D se dodaju njihovim vrijednostima prije početka obrade bloka i prosljeđuju dalje ako ima još blokova. Ukoliko nema više blokova, riječi se spajaju nazad u 128 bitni ključ i vraćaju prikazane kao mali endian. Ključ se obično ispisuje u heksadekadskom obliku s 32 znakova.

## 2.3 SHA-1

## 2.3.1 Povijest

SHA-1 je 1993. u SAD-u razvijen kao dio projekta „Capstone project“. Prvobitno je napravljen SHA-0 koji je radi mane u sigurnosti dorađen i 1995. g. postaje SHA-1. SHA-1 se danas smatra nesigurnim za službenu upotrebu u enkripciji. Prva realna demonstracija napada na puni SHA-1 je izveden 2015. i procijenjena cijena je od US$75k do $120k, što je unutar budžeta tada kriminalnih i vladinih organizacija. U 2017. svi veći web preglednici prestaju prihvaćati SHA-1 SSL certifikate. Od 2020. postoje praktični napadi i u istoj godini je Windows prekinuo podršku za potpis koda (code signing).

## 2.3.2 Upotreba

SHA-1 se danas koristi u kontroli verzije. Za to ga koriste sustavi kao Git, Mercurial i Monotone. Za enkripcijske funkcije preferiraju SHA-2 i novije algoritme. Jedino drugo mjesto gdje se koristi je za pohranu lozinki gdje se MD5 smatra preslabim.

## 2.3.3 Algoritam

A screenshot of a game

Description automatically generated

Slika : Shema za SHA1

Slika 2 prikazuje shemu rada za SHA-1. SHA-1 radi sa 512 bita. Nadopuna je ista kao u MD5 s jednom razlikom. U SHA-1 veličina originalne poruke u bitovima je izražena kao veliki endian. Sve vrijednosti su veliki endian. SHA-1 ima ključ od 160 bita koje dijeli u 5 riječi veličine 32 bita. To su A, B, C, D i E. Inicijalizirane su na određenu početnu vrijednost. Iz poruke se uzimaju blokovi od 512 bita i počinje obrada. Blok od 512 bita se dijeli na 16 riječi Wt od 32 bita koje se proširuju na 80 riječi. Obrada se izvršava u 80 runde za svaki blok. U 80 rundi razlikujemo 4 grupe po 20 rundi u kojima se mijenjaju formule i konstante Kt. Kroz 80 runde se također mijenjaju 32 bitne riječi Wt. Lijeva rotacija (<<<) bitova se odvija nad riječi A i B. Sva se zbrajanja () odvijaju po modulu 2^32.

Formule:

runda 1-20

runda 21-40

runda 41-60

runda 61-80

Nakon formule riječi A,B,C,D i E se spremaju jedno u drugu kao prikazano na shemi. Poslije svih 80 rundi vrijednosti A,B,C,D i E se dodaju njihovim vrijednostima prije početka obrade bloka i prosljeđuju dalje ako ima još blokova. Ukoliko nema više blokova, riječi se spajaju nazad u 160 bitni ključ i vraćaju prikazane kao veliki endian. Ključ se obično ispisuje u heksadekadskom obliku sa 40 znakova.

## 2.4 SHA-256

## 2.4.1 Povijest

SHA-256 spada u grupu sažetaka SHA-2. SHA-2 je NSA ( National Security Agency) dizajnirala i objavila 2001. u SAD-u . Tu spadaju SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/384. Nazvani su po duljini poruke u bitovima. Do danas SHA-256 je siguran od napada.

## 2.4.2 Upotreba

SHA-256 se danas koristi u mnogim aplikacijama i protokolima; TLS, SSL, PGP, SSH, S/MIME i IPsec. SHA-256 se također koristi za autentifikaciju programskih paketa u Debian-u ( otvoreni OS baziran na Linux jezgri) i standard potpisa poruka DKIM ( DomainKeys Identified Mail) koji koriste mail poslužitelji kao npr. Gmail. Linux i Unix koriste SHA-256 i SHA-512 za sažimanje lozinki. U kripto valutama se koristi za potvrdu transakcija i računanje PoS ( Proof of Stake) i PoW (Proof of Work). Posljednja upotreba je u zaštiti klasificiranih podataka na sistemima novijima od WindowsXP SP2 za koje nema podršku. Na njegovom mjestu se još koristi SHA-1.

## 2.4.3 Algoritam

A screenshot of a computer

Description automatically generated

Slika : Shema za SHA-256

Slika 3 prikazuje shemu rada za SHA-256. SHA-256 radi sa 512 bita. Nadopuna je ista kao u SHA-1. Sve vrijednosti su veliki endian. SHA-256 ima ključ od 256 bita koje dijeli u 8 riječi veličine 32 bita. To su A, B, C, D , E, F, G i H. Inicijalizirane su na određenu početnu vrijednost. Iz poruke se uzimaju blokovi od 512 bita i počinje obrada. Blok od 512 bita se dijeli na 16 riječi Wt od 32 bita koje se proširuju na 64 riječi. Obrada se izvršava u 64 runde za svaki blok. U 64 rundi se mijenjaju konstante Kt i riječi Wt. Desna rotacija (>>>) bitova se odvija nad riječi A i E. Sva se zbrajanja () odvijaju po modulu 2^32.

Formule:

Nakon formule riječi A, B, C, D, E, F, G i H se spremaju jedno u drugu kao prikazano na shemi. Poslije svih 64 rundi vrijednosti A, B, C, D, E, F, G i H se dodaju njihovim vrijednostima prije početka obrade bloka i prosljeđuju dalje ako ima još blokova. Ukoliko nema više blokova, riječi se spajaju nazad u 256 bitni ključ i vraćaju prikazane kao veliki endian. Ključ se obično ispisuje u heksadekadskom obliku sa 64 znakova.

## 2.5 Sličnosti i razlike

|  |  |  |  |
| --- | --- | --- | --- |
|  | MD5 | SHA-1 | SHA-256 |
| Veličina bloka (bit) | 512 | 512 | 512 |
| Veličina dopune |  |  |  |
| Runde | 64 | 80 | 64 |
| Ki(Konstante) | 64 | 4 | 64 |
| Wt/Mi veličina (bit) | 32 | 32 | 32 |
| Wt/Mi proširene | 16/ne | 16/80 | 16/64 |
| Ključ (bit) | 128 | 160 | 256 |
| Mali/veliki endian | mali | veliki | veliki |
| Ključ; broj riječi | A, B, C, D |  |  |
| Sigurnost šifriranja | probijen | probijen | siguran |

Tablica : Usporedba metoda MD5, SHA-1 i SHA-256

Sva 3 sažetka rade sa jednakim blokovima od 512 bita.

Dopune uvijek dodaju 1 bit i ostavljaju 64 bita za duljinu poruke, prije čega se po potrebi popunjava nulama. Razlika je što MD5 izražava duljinu originalne poruke u malom endianu, a SHA-1 i SHA-256 u velikom endianu.

Broj rundi nema neko posebno značenje nego balansira između brzine i sigurnosti sažimanja.

Broj različitih konstanti Ki ukazuje da je vraćanje korijenima najsigurnija opcija kada se pojavi slabost u novoj generaciji. Također način na koji se dobiva vrijednost je bitna.

MD5: , i = od 0 do 63

SHA-1: , x = { 2, 3, 5, 10}

SHA-256: , decimalni dio, x = prva 64 primarna broja (od 2 do 311)

Sva 3 sažetka dijele blok na riječi od 32 bita što im služi kao osnova za obradu sažetka i optimalni format ostalih varijabli i konstanti.

SHA-1 i SHA-256 proširuju 16 riječi na respektivni broj rundi s tim da su one kao prvih 16 članova nepromijenjene.

Veličina ključa direktno prikazuje koliko ćemo imati znakova u heksadekadskom prikazu. Za MD5 32, SHA-1 40 i SHA-256 64 znakova.

U MD5 su sve vrijednosti izražene u obliku mali endian. SHA-1 i SHA-256 koriste veliki endian,

Ključ riječi su podijeljene na članove A, B, C, D itd. U slučaju MD5 i SHA-1(dole lijevo) su identične uz to da SHA1 ima dodanog člana E pošto ima veći ključ. SHA-256 (dole desno) uzima prva 32 bita frakcijskog dijela od korijena prvih 8 primarnih brojeva od 2 do 19.

A = 0x67452301 A = 0x6A09E667 F = 0x9B05688C

B = 0xEFCDAB89 B = 0xBB67AE85 G = 0x1F83D9AB

C = 0x98BADCFE C = 0x3C6EF372 H = 0x5BE0CD19

D = 0x10325476 D = 0xA54FF53A

E = 0xC3D2E1F0 E = 0x510E527F

Sve ove vrijednosti se nazivaju “Nothing up my sleeve numbers”. Takvi brojevi se dobivaju od jednostavnih formula ili brojeva. Npr. A u MD5 je mali endian od 0x01234567. SHA-256 koristi formulu koju svako može provjeriti i dobiti iste početne varijable. Prozirnost tih vrijednosti pokazuje da nema skrivenih namjera ili programa za neovlašten ulaz (backdoor) u algoritmu.

MD5 i SHA-1 su probijeni što ih čini nesigurnim za upotrebu u šifriranju. Kao obični sažetci za provjeru integriteta podataka i eventualno pohranu lozinki iako nije preporučeno. SHA-256 zato uz novije sažetke preuzima njihovu ulogu u zaštiti podataka.

# Implementacija u .NET Frameworku

## 3.1 Dostupne implementacije unutar .NET Frameworka

Klase iz Cng (Cryptography Next Generation) se koriste isključivo za sažimanje, ne pružaju šifriranje ili dešifriranje.

Klase implementirane u Crypto Service Provider i Managed pružaju šifriranje i dešifriranje.

Sve navedene klase vraćaju vrijednost u obliku niza bajtova. Za prikaz u heksadekadskom obliku je potrebno obaviti pretvorbu ili prekoračiti metodu ComputeHash.

## 3.2 Aplikacija

U ovom radu implementacija je odrađena u obliku windows aplikacije. Aplikacija se sastoji od 3 prozora: glavni izbornik, hasher i test.

## 3.2.1 Glavni izbornik

Glavni izbornik prikazan na slici 4 se otvara prilikom pokretanja aplikacije i dočekuje korisnika. Ima 2 gumba „Hasher“ i „Test“ koji otvaraju svoje respektivne prozore.

A screenshot of a computer

Description automatically generated

Slika : Glavni izbornik

## 3.2.2 Hasher

Hasher obrađuje tekst i proizvodi sažetak za Cng implementacije MD5, SHA-1 i SHA-256 unutar .NET Frameworka. Uz njih dolazi pojedinačna originalna implementacija. Cng implementacije služe kao provjera valjanosti originalnih implementacija. Unos teksta je moguć na 2 načina. Prvi je direktni upis u ponuđeni prozor. Drugi je pritiskom na gumb „Pronađi datoteku“ koji otvara dijalog u kojem možete odabrati željenu „.txt“ datoteku. Drugi način automatski izvodi sažimanje i izbacuje rezultat. Za prvi način je moguće pozvati svaku metodu sažimanja pojedinačno ili sve odjednom klikom na gumb „Pokreni Sve“ . Pošto za svaki sažetak imamo 2 implementacije, pojedinačni poziv omogućuje usporedbu rezultata za različite unose. Ako to nije dovoljno pored svakog rezultata postoji gumb „Kopiraj u Međuspremnik“ koji sprema rezultat u međuspremnik. Radio gumbi ograničuju upotrebu samo jednog načina unosa teksta u određenom trenutku. Na kraju gumb „Natrag“ nas vraća na glavni izbornik. Na slikama 5, 6, 7 i 8 se vidi primjer svega navedenog.

A screenshot of a computer

Description automatically generated

Slika : Prozor Hasher

A screenshot of a computer

Description automatically generated

Slika : Rezultat kada je unos prazan odnosno „“

A screenshot of a computer

Description automatically generated

Slika : Rezultat kada je unos „a“

A screenshot of a computer

Description automatically generated

Slika : Razlika između rezultata unosa „a“ ( prva 3) i unosa „ab“ (zadnja 3)

3.2.3 Test brzine

Test brzine služi za usporedbu brzine sažimanja. Popis za provjeru vam omogućuje odabir sažetaka koje želite uspoređivati. Jedan sam za sebe, nasumičnu skupinu ili sve odjednom. Prilikom otvaranja graf prikazan na slici 9 služi kao rezervirano mjesto prije prvog pokretanja. Pravi graf je malo drugačiji u paleti boja i rezultatu. Klikom na gumb „Datoteka“ otvara se dijalog za pretragu „.txt“ datoteka u kojem odaberite željenu datoteku. Nakon odabira je gumb „Pokreni“ omogućen. On pokreće odabrane metode i prikazuje ih u grafu. Vrijednosti su izražene u milisekundama. U legendi su prikazani ime i boja kojem je prikazan koji sažetak u grafu. Gumb „Natrag“ nas vraća na glavni izbornik.A screenshot of a graph

Description automatically generated

Slika : Prozor Test brzine

## 3.2.4 Brzina sažimanja

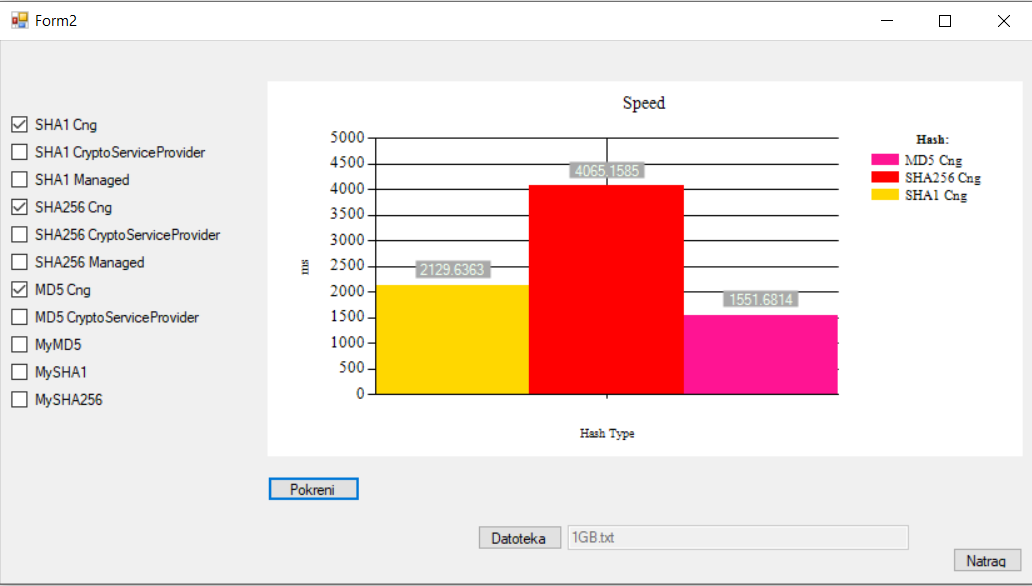
Osim sigurnosti i preciznosti postupke sažimanja možete usporediti u brzini izvođenja za različite količine podataka. Rezultati će uvijek varirati ovisno o hardveru koji se koristi, ali za usporedbu vam je bitno da se izvode na istom računalu. Iz slike 10 vidimo da MD5 ima prednost s manjom količinom podataka npr. obična lozinka od 8 znakova. SHA-1 je uz njega dok SHA-256 treba više nego duplo vremena za isti zadatak.

A screenshot of a computer

Description automatically generated

Slika : Usporedba brzine 3 sažetka u obradi lozinke od 8 znakova

Slika 11 prikazuje rezultat nakon obrade 1GB podataka, što predstavlja malo ozbiljniji pothvat. Sada se bolje vidi da je MD5 brži od SHA-1, dok SHA-256 radi polako, ali sigurno. Ovo predstavlja jedni razlog zašto se MD5 uopće može naći u današnjim sustavima.



Slika : Usporedba brzine 3 sažetka u obradi 1GB podataka

Svaki sažetak ima različite implementacije što može utjecati na brzinu izvedbe. Na slici 12 vidimo sve zadano dostupne implementacije unutar .NET Frameworka. Managed implementacije se ističu kao spore u odnosu na druge biblioteke. Odnos između brzine MD5, SHA-1 i SHA-256 ostaje ravnomjeran kroz različite implementacije.

A screenshot of a computer

Description automatically generated

Slika : Usporedba brzine svih zadanih sažetaka u obradi 1GB podataka

Na slici 13 vidimo usporedbu originalnih implementacija u odnosu na postojeće. Vidljiva je visoka razina optimizacije i kako prezentacija žrtvuje brzinu izvedbe.

A screenshot of a computer

Description automatically generated

Slika : Usporedba brzine originalnih sažetaka sa "Crypto Service Provider" sažecima nad 1MB podataka

# PRAKTIČNI RAD-PRIKAZ U SLIKAMA I TABLICAMA DIJELOVA IZRADE ZAVRŠNOG I DIPLOMSKOG RADA

Za praktični dio nam je važan kod. Kod se dijeli u dvije glavne skupine: sučelje (frontend) i pozadine (backend). U frontendu povezujemo prozore, pozivamo dijaloge, prikazujemo podatke i pozivamo metode iz backenda.

## 4.1 Frontend

## 4.1.2 Prozor „Hasher

Na slici 14 vidimo dvije metode. ConvertStringToByteArray() uzima unos korisnika i vraća red bajtova. Druga metoda ConvertByteArrayToString() uzima red bajtova i vraća niz malih slova.

A screen shot of a computer program

Description automatically generated

Slika : Pomoćne metode za pretvorbu iz reda bajtova u niz i obrnuto

## 4.1.3 Prozor „Test Brzine“

Na slici 15 je prikazana metoda Hasher(). Ova metoda vraća listu vremena potrebno za izvršenje odabranih metoda sažetaka. Vrijeme je izraženo u milisekundama.

A screen shot of a computer program

Description automatically generated

Slika : Metoda Hasher()

Na slici 16 vidimo kod za klik na gumb „Pokreni“. Gumb „Pokreni“ je najbitniji unutar ovoga prozora. Poziva metode koje mjere brzinu odabranih metode sažimanja nad datotekom i na kraju prikazuje rezultat u grafu (naziv „Pregled“ je promijenjen u „Pokreni“ na frontendu).

A screen shot of a computer code

Description automatically generated

Slika : Klik na gumb „Pokreni“

Na slici 17 je prikazana metoda PickedHash(). U red „a“ spremamo bool vrijednost odabranih hashova. Petlja sprema odabrane hasheve u listu cijelih brojeva „picked“ koja istovremeno služi kao povratna vrijednost.

A screen shot of a computer

Description automatically generated

Slika : Metoda PickedHash() vraća listu odabranih sažetaka

Na slici 18 je prikazana metoda HashNames(). Prvo poziva metodu PickedHash() kako bi dobila listu odabranih sažetaka. Iz polja nizova „names“ prebire odabrana i sprema ih u listu „pickednames“. „Pickednames“ se vraća kao povratna vrijednost.

A screen shot of a computer code

Description automatically generated

Slika : Metoda HashNames() vraća listu odabranih sažetaka

## 4.2 Backend

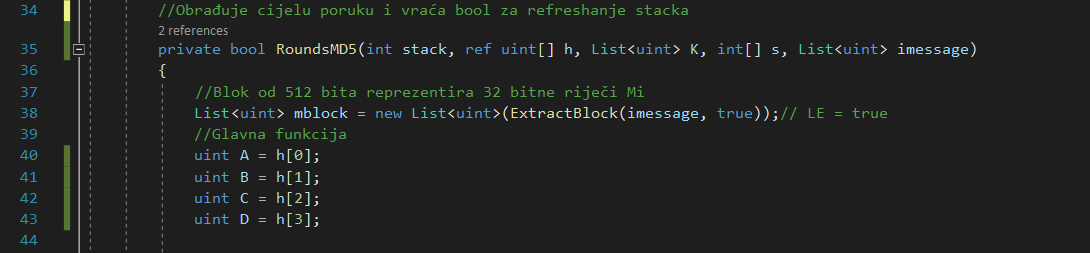
Metoda MyMD5() prikazana u slici 19 implementira sažetak metodu za MD5. Primljeni „array“ se prebacuje u listu „modmessage“ radi lakše obrade u daljnjem izvođenju koda. Metoda AppendPadd() obavlja dopunu. Zatim se obavlja pretvorba liste „modmessage“ u listu „imessage“ koja je tipa uint (cijeli broj bez predznaka). To radimo jer želimo da su nam sve ne indeksne vrijednosti istog tipa u daljnjoj obradi. Vrijednosti A, B, C i D su konstante samo prilikom inicijalizacije. Kasnije se kroz runde mijenjaju i postaju dio ključa. Konstanta „K“ i „s“ su predodređene za ovaj algoritam i mijenjaju se po rundama. Pred kraj je petlja koja poziva metodu RoundsMD5() dok ne obradi „imessage“ do kraja. U RoundsMD5() se prosljeđuju pripremljene konstante i „array“. Ta metoda izvršava glavni dio algoritma. Na kraju metoda MessageDigest() spaja promijenjene vrijednosti A, B, C i D u sažetak oblika mali endian. Ta se vrijednost vraća kao niz.

A computer screen shot of a code

Description automatically generated

Slika : Metoda MyMD5()

Slika 20 prikazuje početak glavnog algoritma za sažetak MD5 u sklopu metode RoundsMD5(). U slici 30 se vidi da se prvo u listu „mblock“ sprema blok od 512 bita kojeg vam iz liste „imessage“ daje metoda ExtractBlock() u obliku mali endian. Nakon toga se rade kopije vrijednosti A, B, C i D.



Slika : Metoda RoundsMD5() prvi dio

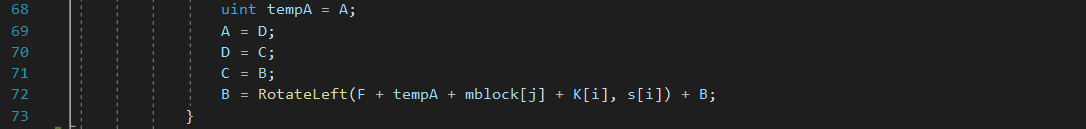
U slici 21 je prikazana glavna petlja u kojoj se obavljaju runde. U slučaju MD5 to je 64 rundi. U svakoj rundi imamo različitu funkciju „F“ i indeks varijablu „j“.

A computer screen shot of numbers

Description automatically generated

Slika : Metoda RoundsMD5() drugi dio

Slika 22 prikazuje drugi dio glavne petlje koji sprema nove vrijednosti varijabli „A“, „B“, „C“ i „D“. Sada se indeks „j“ koristi za odabir riječi unutar bloka „mblock“. Indeks „i“ označava broj rundi i po njemu se odabire konstante „K“ i „s“.



Slika : Metoda RoundsMD5() treći dio

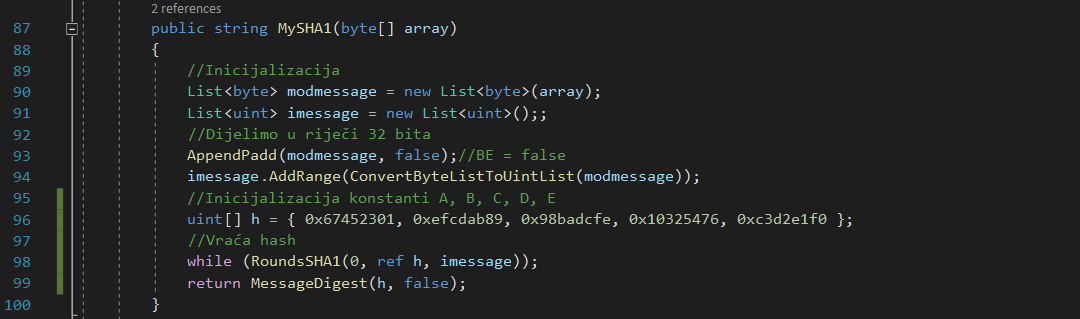
Slika 23 prikazuje drugi dio glavne petlje. Sada van petlje nakon 64 rundi dodajemo rezultat obrade bloka u „h“. Nakon toga provjeravamo je li ostalo još poruke za obradu. Ako to nije slučaj rekurzivno pozivamo metodu gdje su nam nove vrijednosti za „stack“ i „h“. Varijabla „stack“ sprema broj rekurzija i služi za oslobađanje stoga da ne bi došlo do prelijeva.

A black screen with green and white text

Description automatically generated

Slika : Metoda RoundsMD5() četvrti dio

Metoda MySHA-1() prikazana u slici 24 je slična metodi MyMD5() s par razlika. Drugi uvjet u metodi AppendPadd() je false, što označava da je vrijednost izražena u velikom endianu. Druga razlika je manji broj konstanti. Red „h“ ima jednog člana više. Metoda MessageDigest() ima drugi parametar false odnosno vraća veliki endian.



Slika : Metoda MySHA1()

U slici 25 osim standardne inicijalizacije i izvlačenja bloka je dodana petlja. Ta petlja proširuje broj članova odnosno riječi unutar bloka „mblock“ sa 16 na 80, što je broj rundi u glavnoj petlji.

A computer screen with text on it

Description automatically generated

Slika : Metoda RoundsSHA1() prvi dio

U slici 26 je prikazana glavna petlja. Unutar petlje su 4 različite funkcije „F“ i konstante „k“.

A screenshot of a computer code

Description automatically generated

Slika : Metoda RoundsSHA1() drugi dio

U slici 27 je prikazan drugi dio glavne petlje. Riječi iz „mblock“ su odabrane po broju runde “i“. Varijabla „temp“ se koristi za privremeno spremanje vrijednosti „A“.

A black screen with blue and white letters

Description automatically generated

Slika : Metoda RoundsSHA1() treći dio

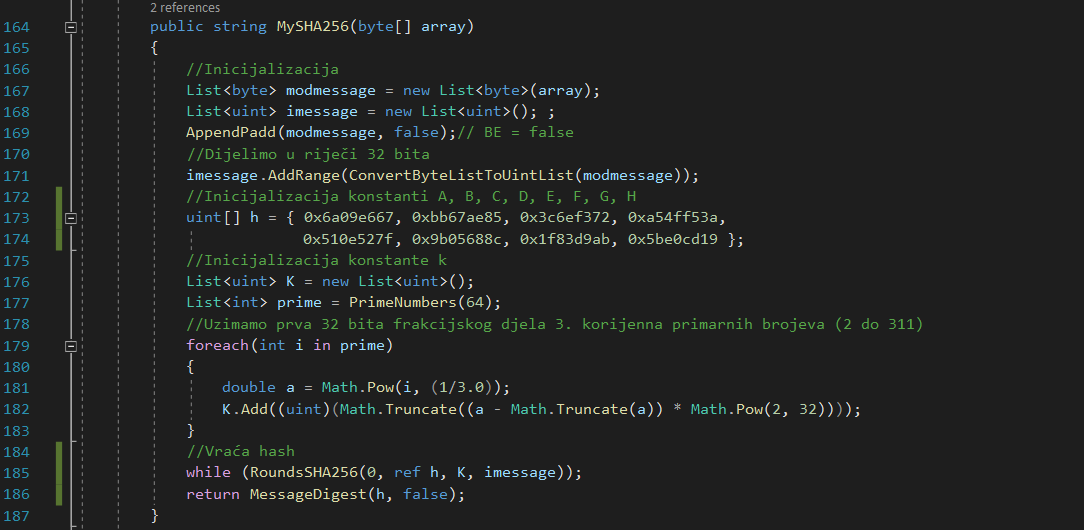
U slici 28 je prikazan zadnji dio metode i istovjetan je za sve implementacije u ovom radu. Razlika je jedino u broju članova „h“.

A computer code on a black background

Description automatically generated

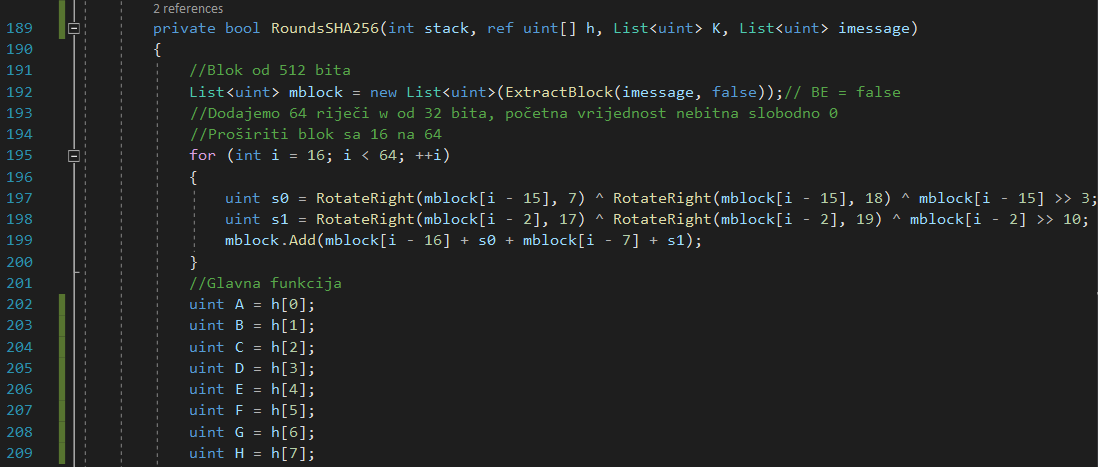
Slika : Metoda RoundsSHA1() četvrti dio

U slici 29 je prikazana metoda MySHA256(). Sve vrijednosti su veliki endian. Red „h“ ima 8 članova. Konstanta „K“ se izračunava po predodređenoj formuli iz prvih 64 primarnih brojeva. Metoda PrimeNumbers() nam daje potrebne primarne brojeve.



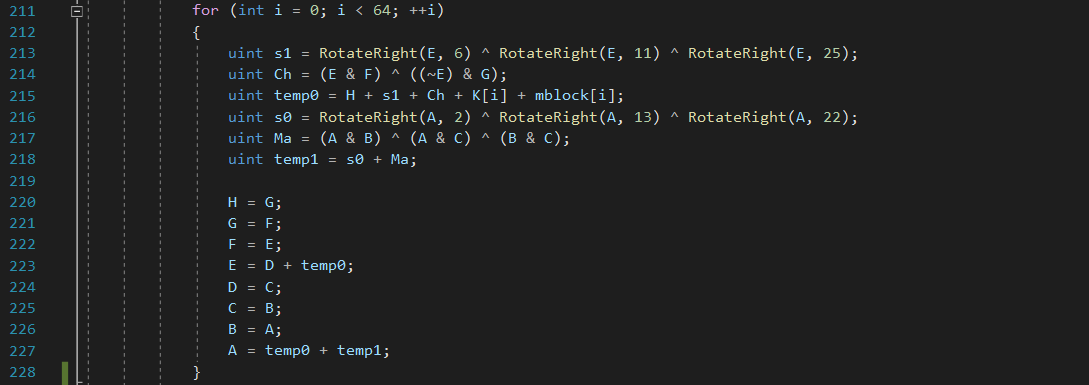
Slika : Metoda MySHA256()

U slici 30 se vidi da metoda RoundsSHA256() također proširuje riječi kao SHA-1 implementacija. Metoda RoundsSHA256() proširuje na totalno 64 riječi. Transformacija riječi je kompleksnija u odnosu na SHA-1 verziju.



Slika : Metoda RoundsSHA256() prvi dio

Na slici 31 vidimo glavnu petlju. U odnosu na MD5 i SHA-1 implementacije. SHA-256 puno više koristi rotaciju bitova. U ovom slučaju udesno pomoću metode RotateRight(). Dvije varijable „temp0“ i „temp1“ se koriste ka pomoć u spremanju rezultata.



Slika : Metoda RoundsSHA256() drugi dio

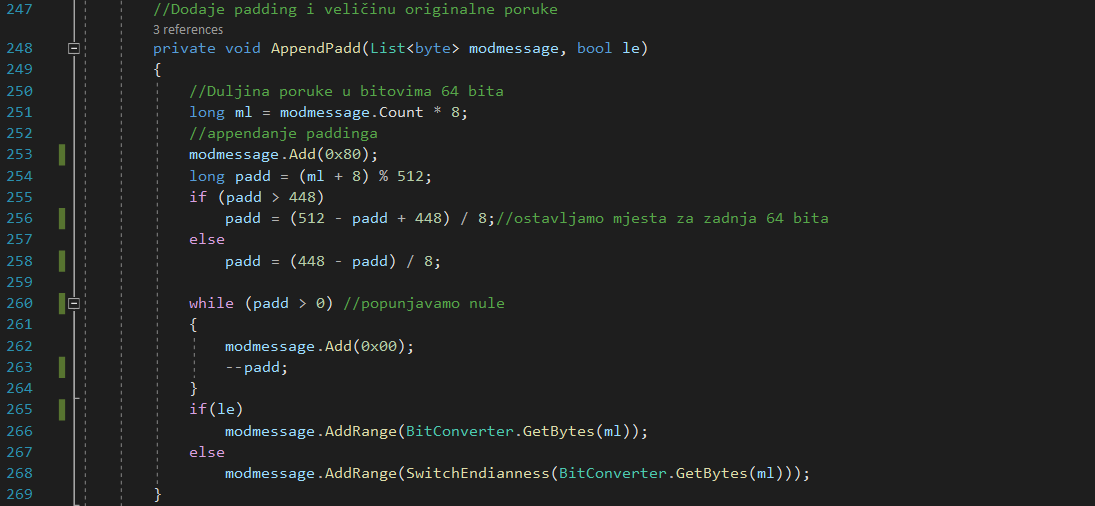
Na slici 32 je kraj metode istovjetan kao u MD5 i SHA-1.

A computer screen with text on it

Description automatically generated

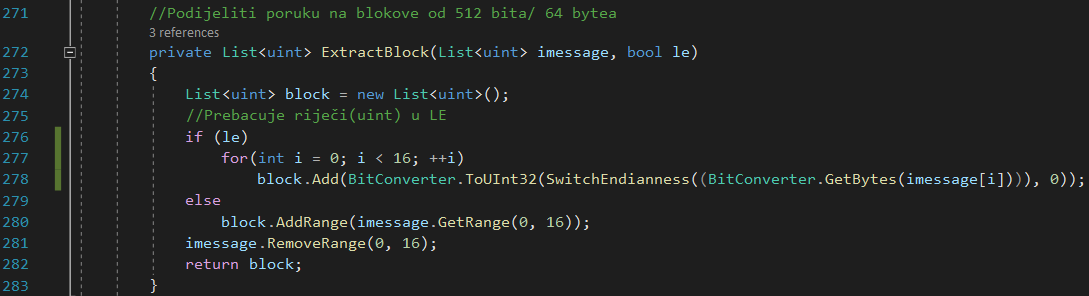
Slika : Metoda RoundsSHA256() treći dio

Na slici 33 je prikazana metoda AppendPadd(). Prvo se u varijablu „ml“ sprema duljina poruke u bitovima. Nakon toga se dodaje bit 1. Zatim računamo veličinu dopune u varijablu „padd“. Petlja puni „modmessage“ nulama. Na kraju se dodaje veličina poruke odnosno „ml“. Način dodavanja ovisi je li se traži mali endian ili veliki endian.



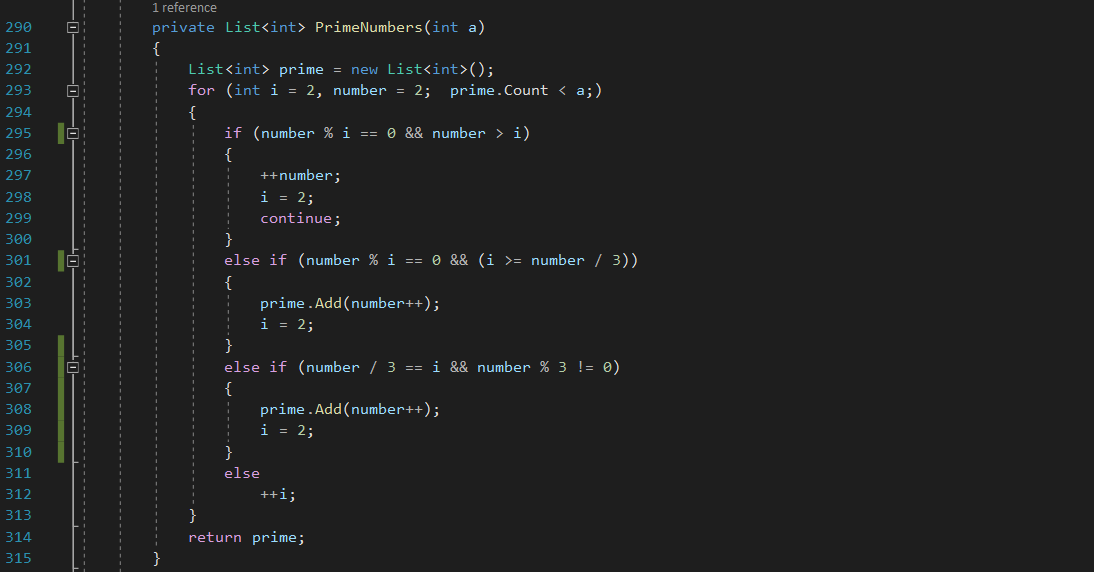
Slika : Metoda AppendPadd()

U slici 34 je prikazane metoda ExtractBlock() koja prati je li pozvana kao mali ili veliki endian. U slučaju mali endian kroz pretvorbe i pomoću metode SwitchEndianness() transformiramo riječ po riječ i dodajemo ju u novi blok. Prije vraćanja novog bloka se taj isti briše iz poruke odnosno liste „imessage“.



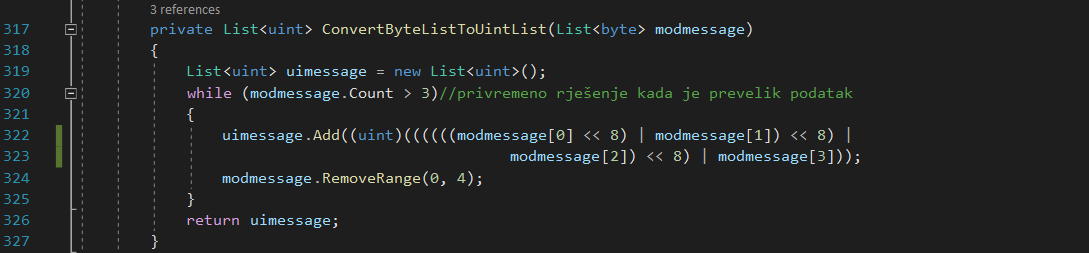
Slika : Metoda ExtractBlock()

Na slici 35 je prikazana metoda PrimeNumbers(). Varijabla „a“ određuje koliko primarnih brojeva želimo pronaći. Unutar petlje imamo 3 uvjeta. Prvi traži je li broj djeljiv s nekim brojem i prelazi na idući broj ako je. Prvi i drugi uvjet uzimaju u obzir male proste brojeve kao 2 i 3. Treći uvjet hvata primarne brojeve s obzirom da su već profiltrirani s prvim i drugim uvjetom. Radi njega drugi uvjet samo hvata vrijednosti 2, 3 i 5. Svi ostali prosti brojevi su uhvaćeni pod trećim uvjetom i smanjuje broj iteracija za otprilike 1/3. Na kraju se vraća lista cijelih brojeva.



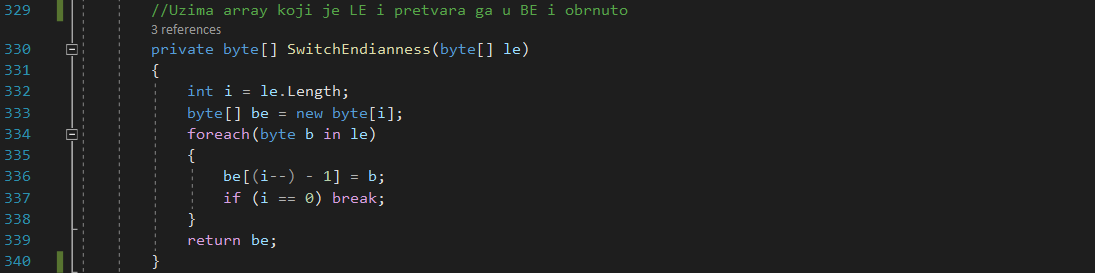
Slika : Metoda PrimeNumbers()

Na slici 36 je prikazana Metoda ConvertByteListToUintList() koja spaja 4 bajta u 1 cijeli broj bez predznaka. Nove vrijednosti su dodane u listu „uimessage“. Lista „modmessage“ se smanjuje svaku iteraciju. Povratna vrijednost je lista cijelih brojeva bez predznaka.



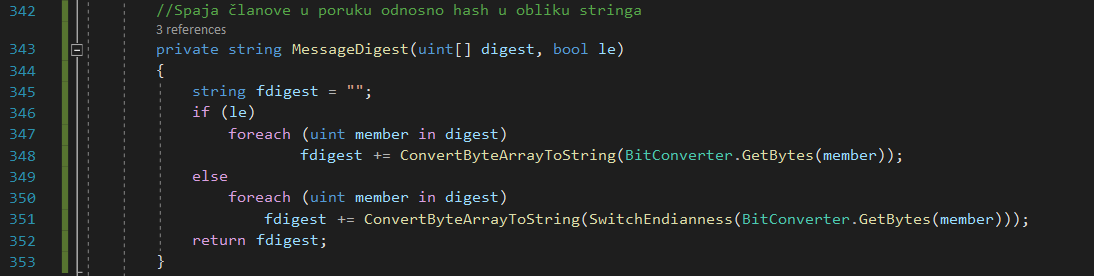
Slika : Metoda ConvertByteListToUintList()

Na slici 37 je prikazana metoda SwitchEndianness(). U indeks „i“ se sprema duljina reda “le“. Stvara se novi red bajtova „be“. Kroz petlju se invertiraju članovi „le“ u „be“. Vraća se red „be“ koji je sada promijenjen sa obrnutim tipom endiana (mali ili veliki).



Slika : Metoda SwitchEndianness()

U slici 38 je prikazana metoda MessageDigest(). Ova metoda spaja riječi, npr. od MD5; A, B, C i D ovdje prikazane kao dio reda „digest“. Ovisno o vrsti endiana koristi drugačije spajanje u niz „fdigest“. Na kraju vraća vrijednost varijable „fidgest“.



Slika : Metoda MessageDigest()

# ZAKLJUČAK

Implementacija sažetaka MD5, SHA-1 i SHA-256 u .Net Frameworku i bilo kojem drugom šta se toga tiče ima jednu veliku prepreku. Testiranje je skoro nemoguće jer najmanja greška u implementaciji algoritma sažimanja daje potpuno drugačiji ključ. Zato kod treba biti čitak da se jasno vidi koji dio koda točno implementira algoritam. Neke vrijednosti i dijelovi rezultata se mogu predvidjeti u cijelosti ili djelomično. U pojedinim dijelovima moguće je precizirati metode dok ne dobijemo predviđeni rezultat, a za neke samo dio rezultata.

Neke metode nisu dio .Net Frameworka te je potrebno stvaranje novih metoda za neke jednostavnije operacije kao rotacija bitova (metoda za pomak bitova postoji) i pretvorba iz malog u veliki endian.

Unatoč svemu MD5, SHA-1 i SHA-256 međusobno dijele dovoljno sličnosti koje omogućuju uniformnu implementaciju. Ovaj način usporava izvedbu programa u zamjenu za preglednost i lakšu usporedbu sažetaka.

Analizom sličnosti i razlika vidimo evoluciju sažetaka. Počinjemo od jednostavnih formula i sigurnosnih mana MD5. Zatim SHA-1 ispravlja greške i uvodi nove ideje i pojačava sigurnost. Na kraju uzimajući dobre i izbacivši loše dijelove MD5 i SHA-1 nastaje SHA-256 koji je dovoljno robustan da još stoji s nama kao standard dok je SHA-3 već razvijen.

Brzina izvedbe pojedinih sažetaka nad manjim dokumentima zadržava proporcionalnost i tijekom obrade većih dokumenata. MD5 je najbrži, zatim SHA-1 i posljednji SHA-256. Kod većih dokumenata postoji šansa za ponavljanje ključa uz slučajne ili namjerne promjene u podacima. Sama veličina ključa direktno utječe na tu šansu. Radi praktičnosti ključ je i dalje relativno mali iako bi se dalo napraviti sažetak koji daje ključ od 64MB. Samo što bi za takvu implementaciju prvo trebalo proizvesti algoritam i sigurnosne provjere nad njim. Izvedba i pohrana ključeva takvih sažetaka ne bi bila praktična. Praktički bi trebali koristiti drugi sažetak samo da provjerite valjanost ključa. Tako da je optimizacija između programa i hardvera bitna u razvoju i implementaciji sažetka.

Osobno mislim da bi svaka web stranica trebala čuvati lozinke sa boljim sažetcima kao SHA-256 radi zaštite korisnika. Većina lozinki se koristi na više mjesta i/ili su slabe. Kada se dođe do maila korisnika napadač ima nadzor i kontrolu nad svim uslugama korisnika ukoliko korisnik ne reagira na vrijeme i promijeni lozinku. Pošto nakon krađe lozinke promjena na novu zahtjeva čovjeka odnosno ručni unos korisnika. U većini slučajeva postoji vremenski odmak djelovanja koji omogućuje intervenciju korisnika. Brza promjena lozinke na sigurniju da se ne ponovi bi trebala spasiti stvar.

Sistemi koji ne podržavaju SHA-256 su odavno izgubili podršku od Microsofta. Noviji Windows 10 je već odbacio SHA-1 2020. godine što je još jedan razlog za prijelaz na SHA-256. Jer ako im SHA-1 nije potreban neće se ustručiti da nađu svoje metode probijanja koje će biti ili neće biti objavljene.

Trenutno je MD5 i SHA-1 obitelj sažetaka bez službene podrške. Zadnja verzija koja ih je podržavala je .NET Framework 4.5.1. Od verzije 4.6 fokus je prebačen na SHA-2 obitelj sažetaka. Zadnja verzija 4.8.1 je i dalje fokusirana na SHA-2.

MD5 i dalje ima svoje mjesto kada je potrebna brza i lagana provjera podataka. Dok SHA-1 preuzima istu ulogu u zahtjevnijim slučajevima. SHA-256 preuzima na sebe sve zadatke uključujući provjeru podataka, službeno i neslužbeno. Iako mnogi programi i web-stranice i dalje koriste MD5 i SHA-1 jer nije ažuriran kod ili je ubačen bez mara za sigurnost. Web stranice s zastarjelim certifikatima mogu biti blokirane upozorenjima od strane anti virusa što može odvratiti korisnike od web stranice bez loših namjera.

Na kraju dana svaki sažetak ima svoj primjenu od internih struktura koje su odvojene od direktnog utjecaja interneta do društvenih mreža i internet trgovina koje su na mreži od 0 do 24.

# LITERATURA

[1]Algorithms Notes For Professionals, goalkicker.com

[2Android Notes For Professionals goalkicker.com

[3]CSharp Notes For Professionals, goalkicker.com

[4]DotNETFramework Notes For Professionals, goalkicker.com

[5]https://web.archive.org/web/20160330153520/https://www.staff.science.uu.nl/~werkh108/docs/study/Y5\_07\_08/infocry/project/Cryp08.pdf (pristupljeno 07.03.2023.)

[6]https://learn.microsoft.com/en-us/dotnet/api/system.security.cryptography?view=net-8.0 (pristupljeno 07.03.2023.)

[7] https://www.lipsum.com/ (pristupljeno 19.09.2023.)

[8]https://msit.powerbi.com/view?r=eyJrIjoiODJmYjU4Y2YtM2M0ZC00YzYxLWE1YTktNzFjYmYxNTAxNjQ0IiwidCI6IjcyZjk4OGJmLTg2ZjEtNDFhZi05MWFiLTJkN2NkMDExZGI0NyIsImMiOjV9 (pristupljeno 24.11.2023)

[9] https://www.zemris.fer.hr/~sgros/stuff/rjecnik.shtml (27.11.2023.)

[10] https://en.wikipedia.org/wiki/MD5 (pristupljeno 07.09.2023.)

[11] https://en.wikipedia.org/wiki/SHA-1 (pristupljeno 07.09.2023.)

[12] https://en.wikipedia.org/wiki/SHA-2 (pristupljeno 07.09.2023.)

[13] https://en.wikipedia.org/wiki/Birthday\_attack (pristupljeno 08.09.2023.)

[14] https://en.wikipedia.org/wiki/Birthday\_problem (pristupljeno 08.09.2023.)

[15]https://support.microsoft.com/en-us/topic/-net-framework-retiring-sha-1-content-9750f20d-a9ef-4d43-853f-2075f0a9d7da (pristupljeno 06.07.2024.)

[16]https://venafi.com/blog/21-of-websites-still-use-sha-1-don-t-they-know-it-s-broken/ (pristupljeno 06.07.2024.)

# SAŽETAK

Ovaj rad prikazuje implementaciju 3 različita sažetka: MD5, SHA-1 i SHA-256 unutar .NET Frameworka. Svaki sažetak je zasebno predstavljen. Od povijesti odnosno nastanka do današnje upotrebe. Kada se prelazi sa starijeg na noviji sažetak jasno se vidi napredak u kvaliteti izvedbe i pokrivanja slijepih točaka. Rad je fokusiran na sam algoritam svakog sažetka i njegovu implementaciju. Struktura sažetaka je veoma slična. Neki dijelovi su skoro identični, što je puno bitnije nego što se čini na prvu. Prikazuje kako je osnovna ideja i dalje dobra i da se uz minimalne promijene može postići značajan napredak. Na drugu stranu, neki dijelovi su potpuno izmijenjeni i dodane su dodatne funkcije kako bi se ispravile veće mane. Originalne implementacije su napisane na relativno jednostavan način kako bi se naglasila struktura algoritma i formula korištenih za izvedbu. Sama aplikacija ima dvije funkcije. Prva omogućuje provjeru rezultata sažetaka proizvoljnog teksta ili običnog dokumenta. Prikazane su .Net Framework implementacije uz originalne kao potvrda valjanosti rezultata. Druga funkcija nudi graf s prikazom brzine izvedbe i ignorira sam rezultat sažimanja. Usporedbom svih sažetaka se vidi da su MD5, SHA-1 i SHA-256 proporcionalni u odnosu jedan na drugog u svakoj implementaciji bez obzira na brzinu izvedbe.

**Ključne riječi:**

* Hashing (sažimanje) je postupak kojim se neki skup podataka obrađuje u fiksni naizgled nasumični skup znakova.
* Hash function ili hash (sažetak) je naziv postupka sažimanja, obično se govori o određenom sažetku npr. MD5.
* Ključ je rezultat postupka sažimanja i ima fiksni broj znakova ovisno o kojem se sažetku radi.
* Veliki endian je način pohrane memorije računala od najbitnije vrijednosti na najmanjoj adresi.
* Mali endian je način pohrane memorije računala od najmanje bitne vrijednosti na najmanjoj adresi.

# SUMMARY

This paper shows implementation of 3 different hash functions: MD5, SHA-1 and SHA-256 inside .NET Framework. Every hash function is represented independently. From history and creation to today's use. When moving from older to newer hash you can clearly see the progress in the quality of performance and coverage of blind spots. The work is focused on the algorithm itself of each hash and its implementation. The structure of the hash is very similar. Some parts are nearly identical, which is much more important than it seems at first. It shows how the basic idea is still good and that with minimal changes significant progress can be made. On the other hand, some parts have been completely changed and additional functions have been added to correct major flaws. The original implementations were written in a relatively simple way to emphasize the structure of the algorithm and the formulas used for its implementation. The application itself has two functions. The first one allows checking the results of hashing of an arbitrary text or a plain document. .Net Framework implementations are shown alongside the original ones as confirmation of the validity of the results. The second function provides a graph showing the performance speed and ignores the hashing result itself. A comparison of all hashes shows that MD5, SHA-1 and SHA-256 are proportional to each other in each implementation regardless of performance speed.

**Keywords:**

* Hashing is a procedure in which some set of dana is transformed into a fixed seamlessly random set of characters.
* Hash function or hash is a name for process of hashing, it usually refers to a certain hash e.g. MD5.
* Key is result of hashing and has fixed number of characters depending on which hash it uses.
* Big endian is a way of storing computer memory from the most significant value on the smallest address.
* Small endian is a way of storing computer memory from the least significant value on the smallest address.