LABORATORIJSKA VEŽBA BR. 6

Primena heš funkcija

- Upoznavanje sa heš funkcijama
- Primena heš funkcija
- Testiranje primene različitih heš funkcija

POTREBNA OPREMA

- Računar sa instaliranim Windows operativnim sistemom
- Instalirani programski paket Cryptool

TEORIJSKE OSNOVE

Heš funkcije

Jedan od najvećih problema koji se javlja kod kriptografije sa javnim ključem vezan je za verodostojnost podataka koji su poslati od osobe A osobi B. Naime, koristeći neki od algoritama sa javnim ključem, šifrovali smo poruku i poslali na destinaciju. Osoba B koja je primila šifrat može uz pomoć svog privatnog ključa da prevede taj šifrat u originalnu izvornu poruku, tj. otvoreni tekst. Međutim, postavlja se pitanje kako da osoba B bude sigurna da je ta poruka baš poslata od osobe A tj. pitanje verodostojnosti primljene poruke. Uslovno gledano možemo da kažemo da je svaka poruka koja se šalje šifrovana privatnim ključem osobe koja tu poruku šalje pa je samim tim i definisana verodostojnost poslate poruke. Međutim, kako je ovde cela poruka šifrovana u slučaju većih poruka to može da bude dosta otežavajuća varijanta kod utvrđivanja njene verodostojnosti. Da bi se ovaj problem rešio primenjuje se tehnika poznata kao Digitalni potpis koji predstavlja skraćeni oblik šifrovane poruke privatnim ključem osobe koja šalje tu poruku i koji nedvosmisleno određuje osobu koja je tu poruku poslala. Za tu svrhu korišćene su Heš funkcije koje predstavljaju određene matematičke operacije koje mogu da bilo koji tekst svedu na unapred definisanu veličinu. **Heš** funkcija (*message digest*) predstavlja jednosmernu funkciju koja obrađuje poruku nefiksne dužine, M, a vraća heš fiksne dužine h. Jednosmerna funkcija označava da se šifrovanje teksta vrši samo u jednom smreu tj. nije moguće da se od heš vrednosti u suprotnom smeru dobije original poruka. Heš funkciju možemo predstaviti kao **h = H(M)**. Generalno, funkcija, tj. primena heša je da poruci da jedinstven identifikator koji će nedvosmisleno predstavljati tu poruku (nešto slično kao otisak prsta ili potpis osobe) koja je tu poruku kreirala. Ako osoba A potpiše šifrovanu poruku i pošalje osobi B, osoba B može izračunati heš kako bi dobila M ali samo ako važi: **H(M)=H(M`)**.

Problem sa hešom je taj što postoji efekat kolizije koji dokazuje da postoje dve različite poruke koje će dati isti heš. **M i M` → H(M) = H(M`)**. Takođe postoji i efekat lavine, odnosno lavinski efekat što znači da ako se u originalnoj poruci promeni samo jedan bit, heš te iste poruke će promeniti čak polovinu ukupnog broja bitova u hešu. Heš algoritmi se koriste, pored potpisivanja dokumenata i radi autentifikacije korisnika. Tako se lozinke korisnika na Linux i Windows sistemima čuvaju u obliku heša u datotekama "shadow" (Linux) i u datoteci SAM (Windows). Tako kada se korisnik loguje i kuca svoju lozinku sistem računa heš koji upoređuje sa hešom koji je ranije upamćen u ovim datotekama. Ako je dobijeni heš isti kao heš koji se nalazi u ovima datotekama, korisnik ima legalan pristup računaru tj. lozinka je u redu. Dosta je bitno da se razume princip rada heš funkcija kako bi bilo jasno da verifikacija dokumenata, lozinki na operativnim sistemima, itd. nije postupak dešifrovanja heša jer bi u tom slučaju značilo da ta heš funkcija nije dobra, tj. nema jednosmernu osobinu. Proces verifikacije se obavlja isključivo ponovnim računanjem heša istog dokumenta i upoređivanjem sa već dobijenim hešom.

Primena heš funkcije kod Linux i Windows operativnih sistema

Lozinke na operativnim sistemima se čuvaju na određenim mestima u odgovarajućim datotekama. Tako, Windows operativni sistemi, čuvaju lozinke u SAM datoteci u kojoj su upisani heševi lozinki svih korisnika jednog računara. Na samom početku, korisnik zadaje željenu lozinku kojom će se prijavljivati na sistem i tada sistem računa MD5 heš vrednost ove lozinke i smešta je u datoteku SAM koja se nalazi na putanji: **C:\WINDOWS\system32\config\SAM**. Sledeći put kada se isti korisnik loguje na sistem, ponovo se računa heš vrednost **ukucane** lozinke i upoređuje sa hešom koji definiše lozinku koju je korisnik izabrao na samom početku. Znači, šifre na Windows-u se u obliku heša čuvaju u SAM datoteci, a pri prijavljivanju korisnika na sistem, korisnik upisuje svoju šifru u ***login form*** i sistem automatski računa heš lozinke koju korisnik trenutno unosi u ***login form***. Izračunati heš upoređuje sa hešom koji se nalazi u sistemu, odnosno u SAM datoteci. Ako su heševi identični, sistem dopušta korisniku da pristupi svom nalogu, dok je u suprotnom korisnik obavešten o neispravnosti svoje lozinke.

Šta je glavni problem sa ovom metodom autentifikacije? Vrlo je moguće da postoje dva **različita** niza karaktera koji će provlačenjem kroz heš funkciju dati isti heš! Ovo je i glavni nedostatak heš funkcija i predstavlja problem **kolizije**. Pored ovog problema, jedan od najvećih problema koji narušavaju sigurnost šifara uopšte je ljudski faktor. Šifre zapisane po čitavoj kancelariji, deljenje šifara kolegama sa posla, itd... Postoje i "Administartori" mreža i sistema koji greškom ili neznanjem ostave nalog **Administrator** nezaštićenim, odnosno bez šifre, jer se za svakog korisnika napravi nalog dok se administratorski nalog zanemaruje usled nekorišćenja. Ovom računaru je omogućen pristup jednostavnim navođenjem ***username***-a kao administrator. Kako bi razbili heševe i ukazali na nedostatke pojedinih heš funkcija, kriptoanalitičari koriste razne metode kao što je napad rečnikom. O čemu se radi? Pošto se SAM datoteka čita radi upoređivanja heša unetog na terminalu i onog koji postoji u sistemu, lako je za pretpostaviti da je moguće napraviti posebnu heš vrednost, iako je ona nastala od neke reči koja nije ista kao šifra korisnika računara, koja odgovara vrednosti u SAM datoteci. Na ovaj način bi u login formu pri prijavi na sistem, bilo moguće navesti tu drugu reč kao lozinku i da nas sistem pusti da koristimo nalog kao regularni korisnik, jer obe reči iako različite daju isti heš. Postoje razni alati za spašavanje zaboravljenih lozinki koji rade baš na ovom principu, dakle, čitaju vrednost heša i koristeći reči i fraze koje su smeštene u bazama tih programa. Kada program dođe do reči ili fraze koja daje isti heš kao onaj u sistemu, smatra se da je šifra razbijena. Kod prozora za login na sistem, ne logujete se ni na jedan od naloga već stisnete dva puta za redom kombinaciju tastera **CTRL+ALT+DEL**, primetićete da se izgled login-a promenio. Sada u polje za username kucate "**administrator**" i potvrđujete sa ENTER. Dali ste uspeli? Ovako imate full pristup računaru i njegovim resursima, korisničkim nalozima, itd...

Primenjivani algoritmi kod dobijanja heša

1. Algoritam MD4 - predstavlja jednosmernu heš funkciju koja prizvodi heš od 128 bitova. Ciljevi projektovanja MD4 su:

a. ***Sigurnost*** - računarski je teško pronaći dve poruke koje daju isti heš
b. ***Neposredna sigurnost*** - sigurnost MD4 nije zasnovana na pretpostavci, kao što je teško faktorisanje.
c. ***Brzina*** - MD4 je pogodan za brže softverske realizacije, zasnovan je na jednostavnom skupu manipulacija bitovima na 32-bitnim operandima.
d. ***Jednostavnosti kompaktnost*** - MD4 je jednostavan koliko je moguće bez velikih struktura podataka ili komplikovanog programa. Optimizovan je za mikroprocesore, posebno za intel

2. Algoritam MD5 je poboljšana verzija MD4 algoritma. MD5 je složeniji ali slično organizovan kao i MD4 i daje heš takođe 128 bitova. Na početku šifrovanja ulazni karakteri prolaze kroz neke početne operacije a zatim MD5 obrađuje ulazne podatke u 512-bitnim blokovima podeljenim na 16 32-bitnih podblokova. Izlaz iz algoritma su 4 32-bitna bloka koji se nadovezuju kako bi napravili 128-bitni heš. Otvoreni tekst se najpre dopuni tako da dužina bude za 64 bita kraća od umnoška broja 512. Dopuna je 1 bit na kraju poruke i onoliko nula koliko je potrebno. Zatim se na sve to dodaje 64-bitna reprezentacija dužine poruke pre dodavanja dopunskih bitova. Inicijalizuju se 4 promenljive od po 32b.

A = 0x01234567
B = 0x89abcdef
C = 0xfedcba98
D = x76543210
Zatim ide glavna petlja koja se ponavlja onoliko puta koliko ima 512-bitnih blokova u poruci. Nakon toga ide konverzija: a → A, b → B, c → C, d → D. Osnovna petlja ima 4 runde. Svaka runda koristi drugu operaciju 16 puta. Svaka operacija izvršava nelinearnu funkciju nad 3 od 4 promenljive. Zatim se taj rezultat dodaje 4-oj promenljivoj i podbloku teksta kao i konstanti. Rezultat se rotira desno za promenljiv broj bitova i rezultat se dodaje jednoj od promenljivih. Na kraju, rezultat zamenjuje jednu od promenljivih.

3. Algoritam SHA pravi heš tako što prvo poruku dopuni tako da joj dužina bude umnožak od 512 bitova. Samo dopunjavanje je kao kod MD5, što znači da prvo ide jedinica pa onoliko nula koliko je potrebno da dužina bude za 64 bita kraća od umnopka broja 512 + 64-bitna reprezentacija dužine poruke pre dopunjavanja. Inicijalizuje se 5 promenljivih:
A = 0x67452301
B = 0xefcdab89
C = 0x98badcfe
D = 0x10325476
E = 0xc3d2e1f0
Sada ide glavna petlja koja obrađuje poruku u blokovima od po 512 bitova I izvršava se onoliko puta koliko ima blokova u poruci. Promenljivoj a se kopira vrednost u A, b → B, c → C, d → D, e → E. Glavna petlja ima 4 runde od po 20 operacija, svaka operacija izvršava nelinearnu funkciju nad 3 od 5 promenljivih, zatim se obavlja pomeranje i sabiranje, slično kao kod MD5.

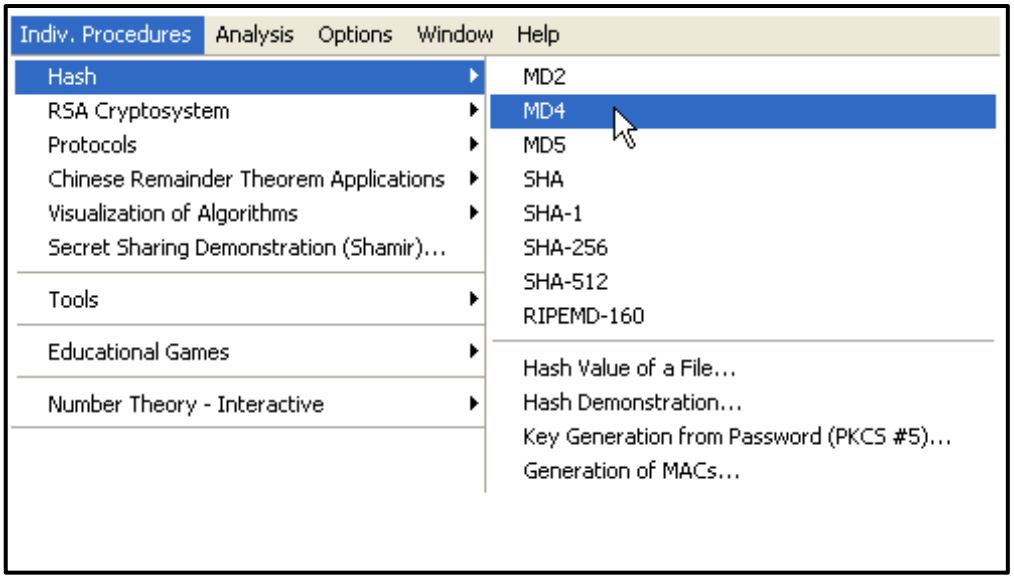
4.Algoritam SHA1 je verzija SHA algoritma koja pravi heš od 160 bita, Algoritam SHA 256 kao i SHA 512 i SHA 384 i SHA 224, proizvode različite dužine heševa. SHA512 → heš=512b, SHA384 → heš=384b,...

5. Algoritam RIPEMD, RIPEMD160 je razvijen za RIPE projekat organizacije European Community. Ovaj algoritam je varijacija algoritma MD4 i dizajniranje da izdrži kriptoanalitičke napade i da napraavi heš od 128 bitova. Razlika između MD4 i RIPE-a je u tome što ovde imamo promenjene rotacije i redosled reči poruke, paralelno se izvršavaju dva primerka algoritma koji se razlikuju samo u konstantama. Nakon svakog bloka izlazi oba primerka se dodaju na promanljive koje se zatim ulančavaju. Verzija RIPEMD160 je algoritam koji daje heš od 160 bitova na izlazu.

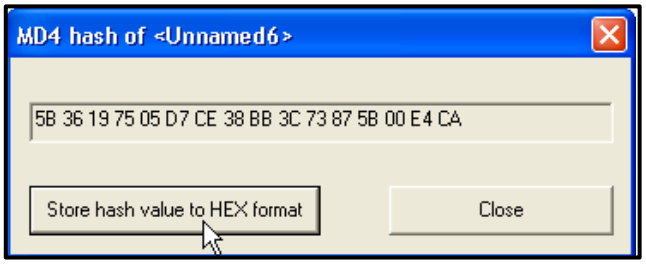
ZADATAK:

1. Demonstracija na Cryptool-u (MD4)

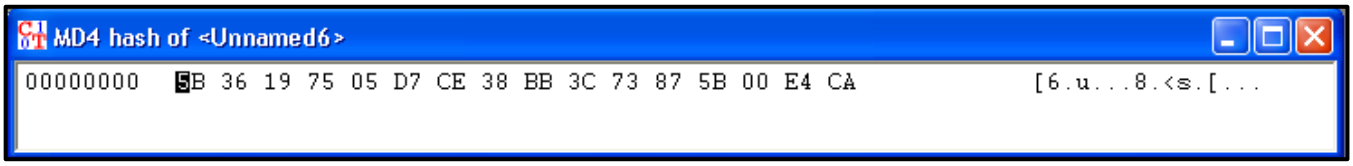
Pokrenite Cryptool i izaberite neki tekst na koji ćete primeniti algoritam MD4 (u našem primeru je izabran tekst Licence Cryptool-a na koju koju ćete MD4 algoritam biti primenjen tj. da bi dobilim odgovarajuću heš vrednost ovog dokumenta). U daljem radu pokazaćemo i uticaj lavinskog efekta koji može da se pojavi kod heš funkcija. Znači, pokrećemo demonstraciju MD4 algoritma koja se nalazi u meniju ***Indiv.Procedures – Hash – MD4*** (kao što je pokazano na donjoj slici):



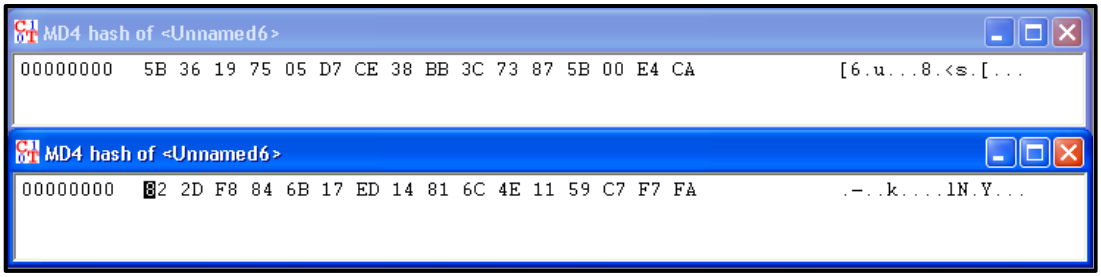
Dobijamo prozor u kome nas Cryptool obaveštava da će heš vrednost prebaciti u heksadecimalni zapis. Klik na ***Store hash value to HEX format:***



Sada imamo przor sa heš vrednošću ovog dokumenta koji je dat u HEX zapisu kao na slici:



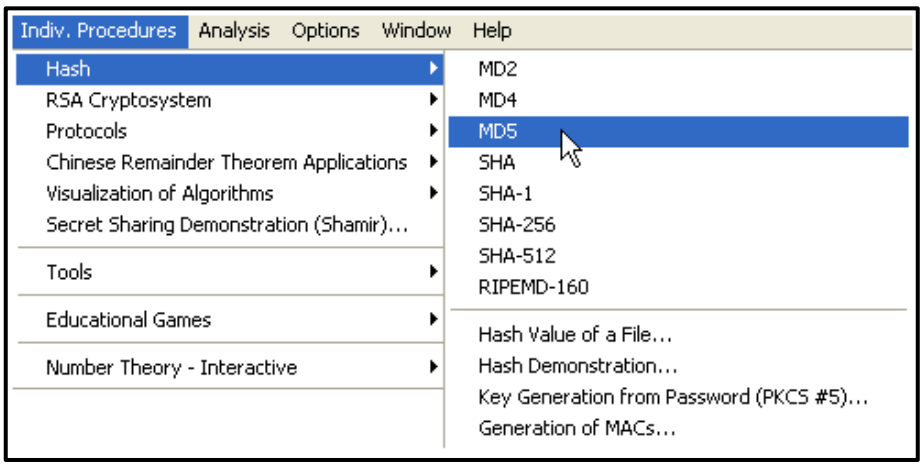
Pošto imamo heš vrednost ovog dokumenta, da bi prikazali uticaj postojanja lavinskog efekta kod heš funkcija, promenićemo u originalnom dokumentu (tekst Licence) samo jedan karakter ili dodati još jedan karakter. Tako možemo na primer da u naslovu teksta, gde stoji "CrypTool", dodamo broj dva, "2" na kraj reda tako da bi sada naslov izgledao ovako: "**CrypTool2**".
Ponovo pokrećemo MD4 algoritam i primenjujemo ga na promenjeni tekst, tj. tekst sa novim naslovom. (CrypTool2). Uporedite rezultate dobijenih heševa, sada i prethodno dobijenog heša koji su prikazani na donjoj slici:



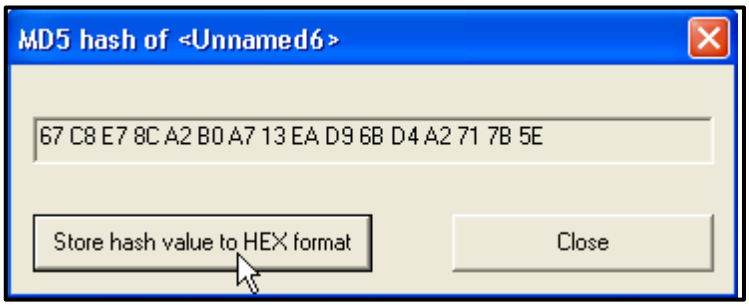
2. Demonstracija na Cryptool-u (MD5)

Pokrenite Cryptool i u novi prozor, takođe kao u prethodnom primeru, uneti neki tekst (u našem slučaju to je opet Licenca Cryptool-a na koji ćetmo primeniti algoritam MD5 kako biste dobili heš vrednost ovog

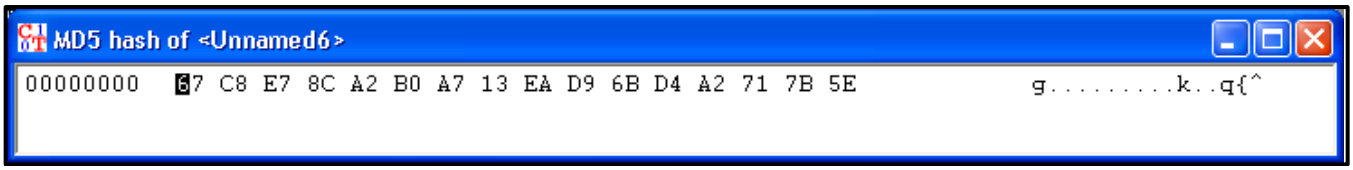
dokumenta. I ovde ćemo prikazati uticaj lavinskog efekta. Znači, pokrećete demonstraciju MD5 algoritma koja se nalazi u ***Indiv.Procedures – Hash – MD5***:



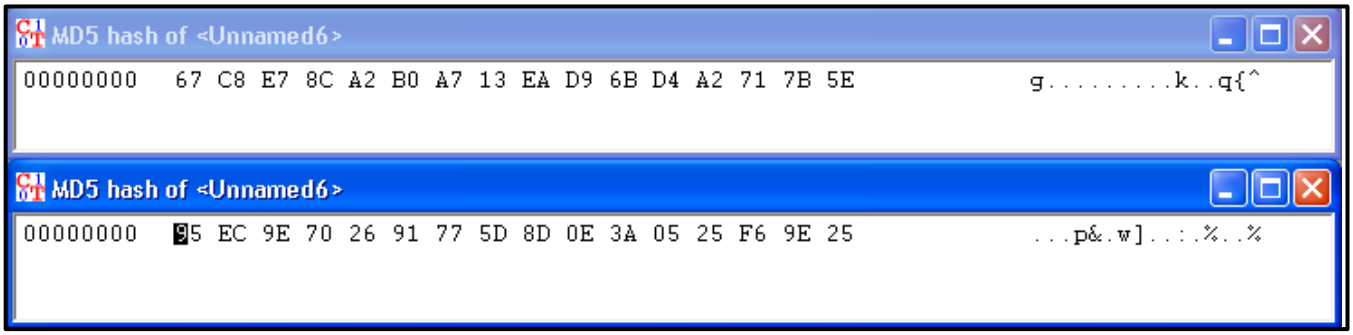
Dobijamo prozor u kom nas Cryptool obaveštava da će heš vrednost prebaciti u heksadecimalni zapis. Klik na ***Store hash value to HEX format***:



Sada imamo przor sa heš vrednošću ovog dokumenta u HEX zapisu:

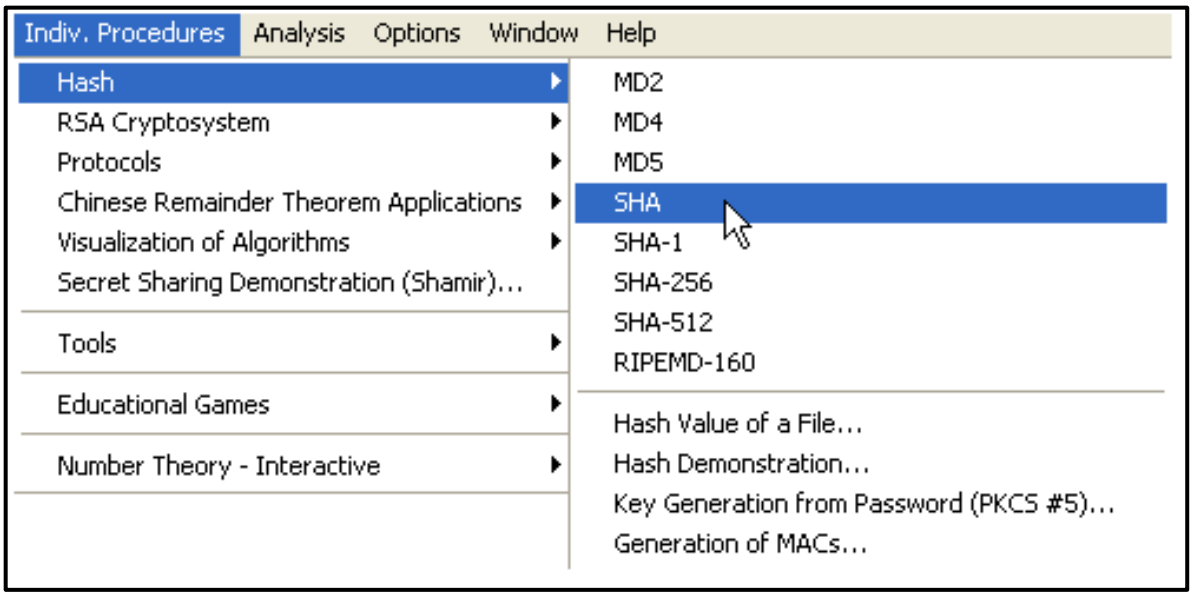


Pošto imamo heš vrednost ovog dokumenta, treba dokazati postojanje lavinskog efekta tako što ćemo u originalnom dokumentu (u našem primeru tekst Licence) promeniti samo jedan karakter ili dodati još jedan karakter. Možete u naslovu teksta, gde stoji "CrypTool", dodati broj dva, "2", na kraj reda kako bi sad naslov izgledao ovako: "**CrypTool2**".
Ponovo pokrećete MD5 algoritam i primenjujete ga na promenjeni tekst, tj. tekst sa novim naslovom. (CrypTool2). Uporedite rezultate dobijenih heševa, sada i prethodno dobijenog heša:

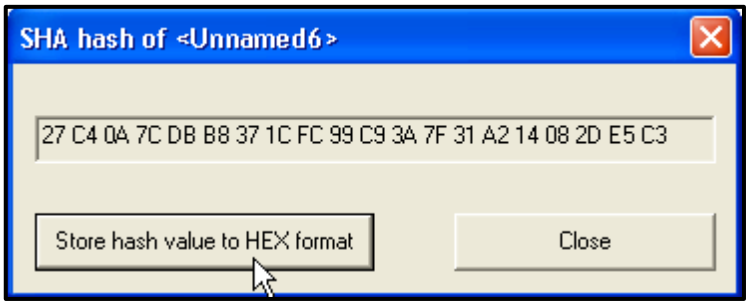


3. Demonstracija na Cryptool-u (SHA)

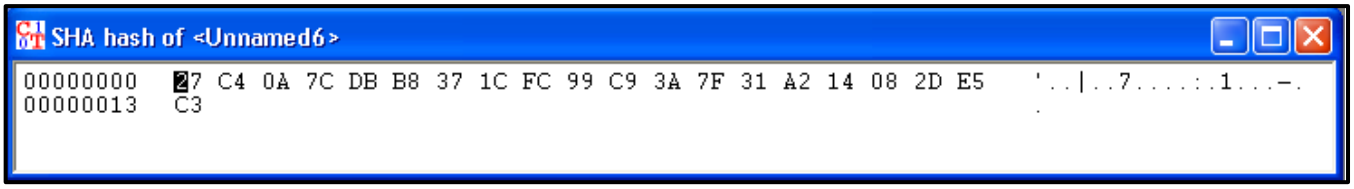
Pokrenite Cryptool i u novi prozor prekopirajte Licencu Cryptool-a na koju koju ćete primeniti algoritam SHA kako biste dobili heš vrednost ovog dokumenta. Znači, pokrećete demonstraciju SHA algoritma koja se nalazi u meniju ***Indiv.Procedures – Hash – SHA***:



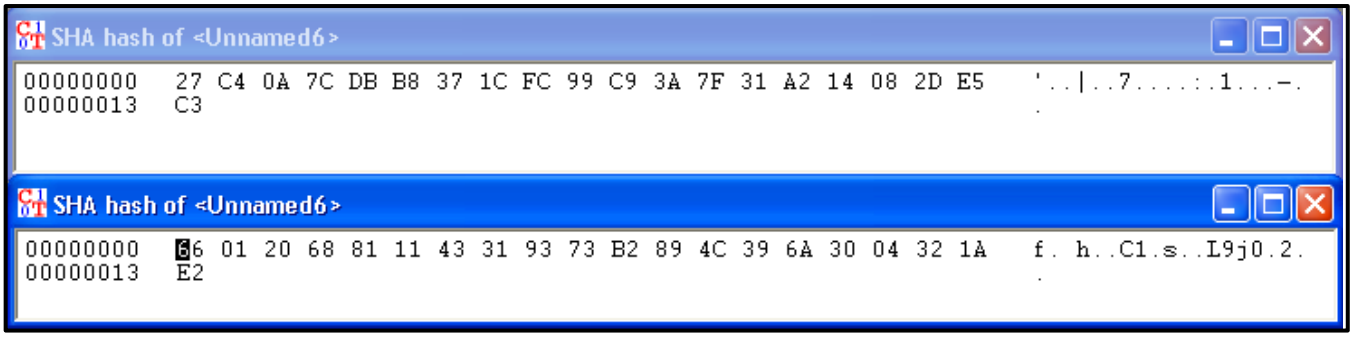
Dobijamo prozor u kom nas Cryptool obaveštava da će heš vrednost prebaciti u heksadecimalni zapis. Klik na ***Store hash value to HEX format***:



Sada imamo przor sa heš vrednosti ovog dokumenta u HEX zapisu:

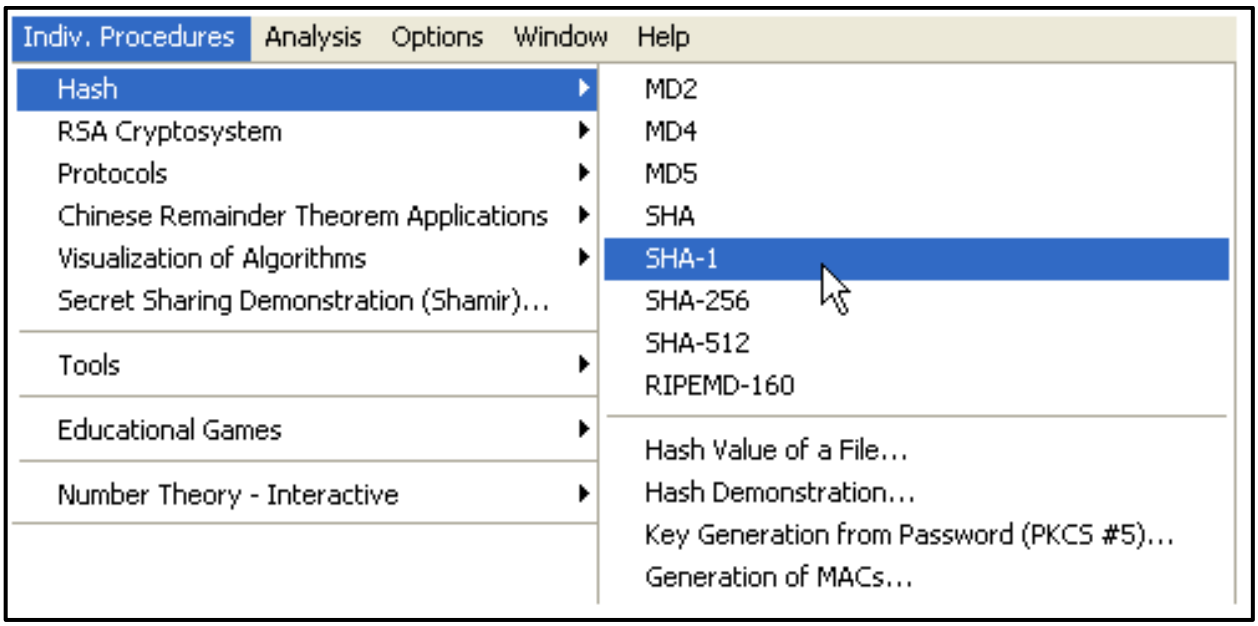


Pošto imamo heš vrednost ovog dokumenta, treba dokazati postojanje lavinskog efekta tako što ćemo u originalnom dokumentu (Licenci) promeniti samo jedan karakter ili dodati još jedan karakter. Možete u naslovu teksta, gde stoji "CrypTool", dodati broj dva, "2", na kraj reda kako bi sad naslov izgledao ovako: "**CrypTool2**". Ponovo pokrećete SHA algoritam i primenjujete ga na promenjeni tekst, tj. tekst sa novim naslovom. (CrypTool2). Uporedite rezultate dobijenih heševa, sada i prethodno dobijenog heša:

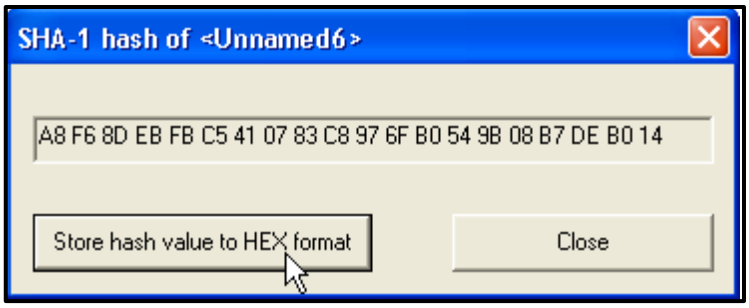


4. Demonstracija na Cryptool-u (SHA-1)

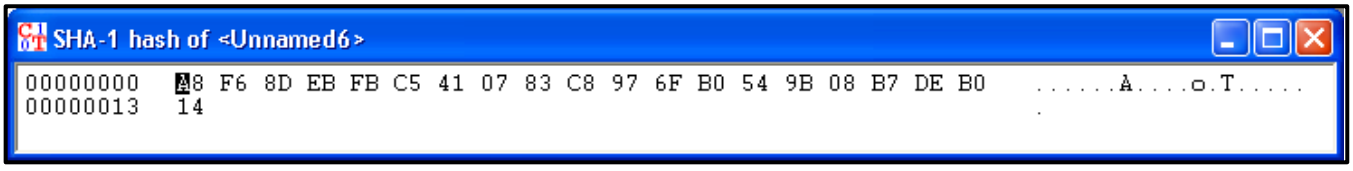
Pokrenite Cryptool i u novi prozor prekopirajte Licencu Cryptool-a na koju koju ćete primeniti algoritam SHA-1 kako biste dobili heš vrednost ovog dokumenta. Znači, pokrećete demonstraciju SHA-1 algoritma koja se nalazi u meniju ***Indiv.Procedures – Hash – SHA-1***:



Dobijamo prozor u kom nas Cryptool obaveštava da će heš vrednost prebaciti u heksadecimalni zapis. Klik na Store hash value to HEX format:

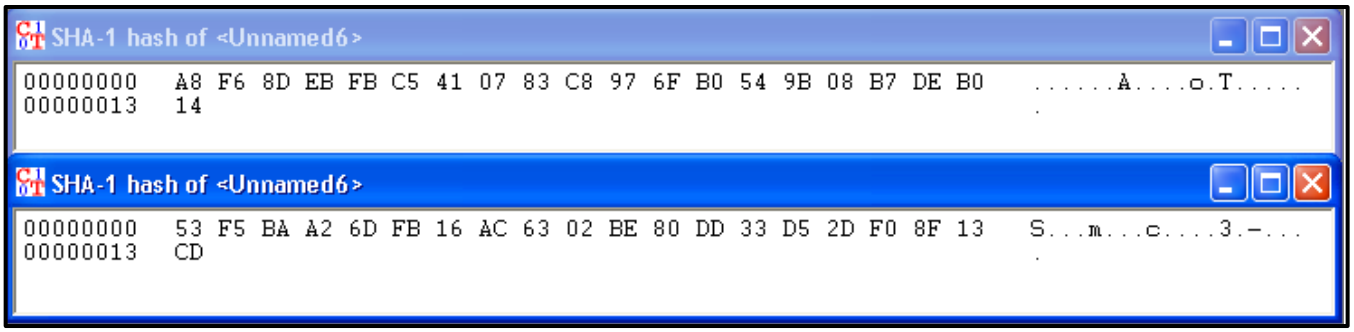


Sada imamo przor sa heš vrednosti ovog dokumenta u HEX zapisu:



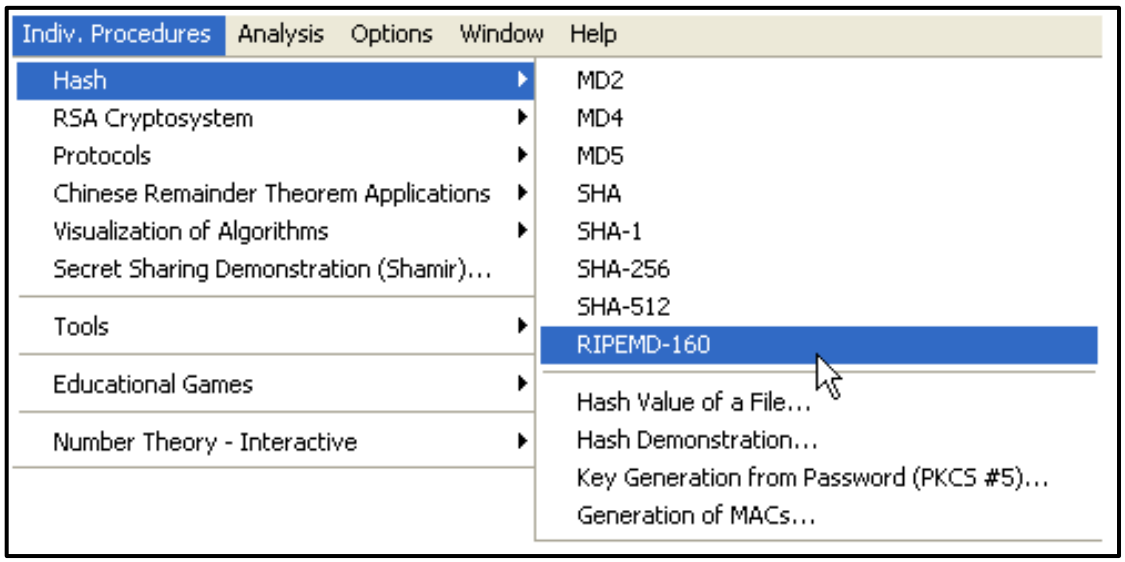
Pošto imamo heš vrednost ovog dokumenta, treba dokazati postojanje lavinskog efekta tako što ćemo u originalnom dokumentu (Licenci) promeniti samo jedan karakter ili dodati još jedan karakter. Možete u

naslovu teksta, gde stoji “CrypTool”, dodati broj dva, “2”, na kraj reda kako bi sad naslov izgledao ovako: “**CrypTool2**”.
Ponovo pokrećete SHA algoritam i primenjujete ga na promenjeni tekst, tj. tekst sa novim naslovom. (CrypTool2). Uporedite rezultate dobijenih heševa, sada i prethodno dobijenog heša:

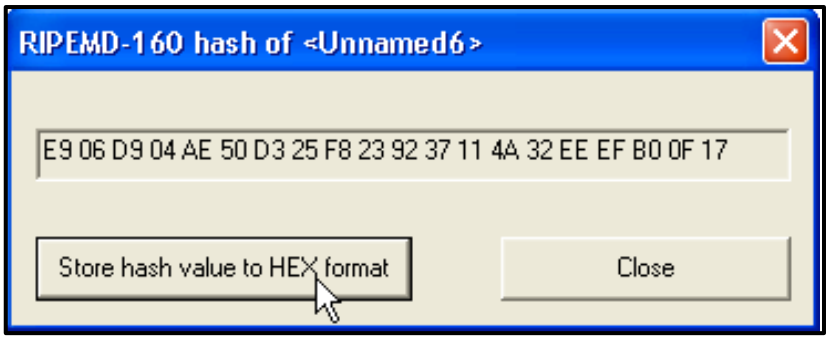


5. Demonstracija na Cryptool-u (RIPEMD)

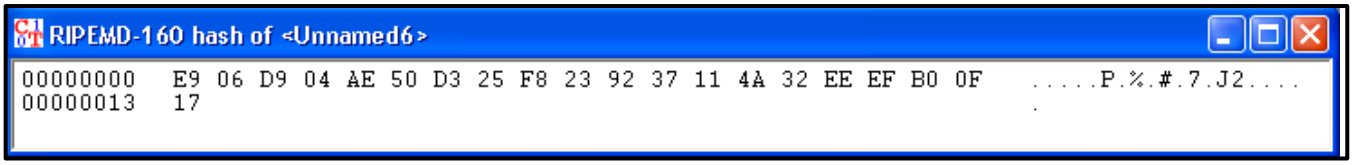
Pokrenite Cryptool i u novi prozor prekopirajte Licencu Cryptool-a na koju koju ćete primeniti algoritam RIPEMD-160 kako biste dobili heš vrednost ovog dokumenta, pokazaćete takođe uticaj lavinskog efekta. Znači, pokrećete demonstraciju RIPEMD-160 algoritma koja se nalazi pod menijem ***Indiv.Procedures – Hash –RIPEMD-160***:



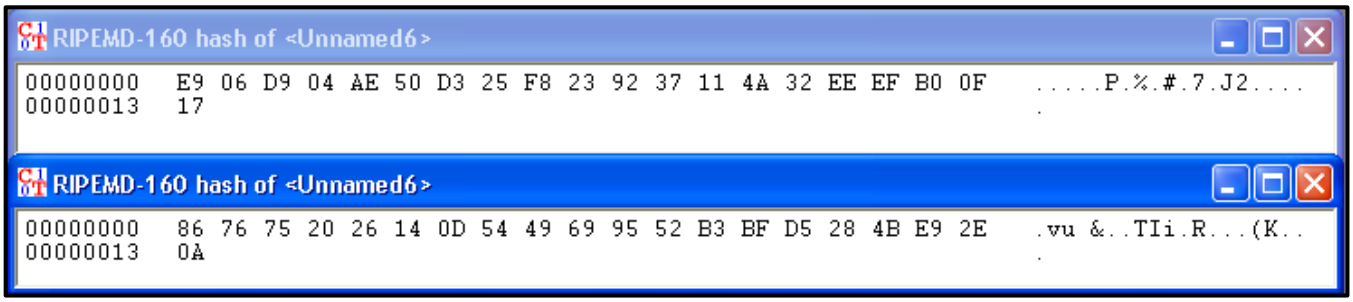
Dobijamo prozor u kome nas Cryptool obaveštava da će heš vrednost prebaciti u heksadecimalni zapis. Klik na Store hash value to HEX format:



Sada imamo przor sa heš vrednosti ovog dokumenta u HEX zapisu:



Pošto imamo heš vrednost ovog dokumenta, treba dokazati postojanje lavinskog efekta tako što ćemo u originalnom dokumentu (Licenci) promeniti samo jedan karakter ili dodati još jedan karakter. Možete u naslovu teksta, gde stoji "CrypTool", dodati broj dva, "2", na kraj reda kako bi sad naslov izgledao ovako: "**CrypTool2**".
Ponovo pokrećete RIPEMD-160 algoritam i primenjujete ga na promenjeni tekst, tj. tekst sa novim naslovom. (CrypTool2). Uporedite rezultate dobijenih heševa, sada I prethodno dobijenog heša:



Pitanja: 1. Šta možemo da zaključimo iz ovih primera?

2. Koliko se heš vrednosti razlikuju ako se primenjuju različiti algoritmi?

Zadatak: Svaki student je dužan da pomoću gore objašnjenih algoritama šifruje **imena i prezimena** svojih članova porodice (Po četiri primera za svaki algoritam, ukoliko je broj članova porodice manji od četiri, studenti uzimaju **ime i prezime svog najboljeg prijatelja** kao četvrti primer).