**Visoka škola računarstva i poslovnih komunikacija eMPIRICA**

**u Brčko distriktu BiH**

**Akademska 2013./2014. godina**

**Studijski program: Inženjerska informatika**

SEMINARSKI RAD

iz predmeta *„PROGRAMIRANJE I PROGRAMSKI JEZICI“*

**IMPLEMENTIRANJE SIGURNOSNIH ALGORITAMA U VISUAL BASIC-U**

STUDENT  
Aleksandar Todorović

brčko, juni 2014. godine

# Uvod

U današnje vrijeme je informacijska sigurnost fokus IT svijeta. Skoro svakodnevno možemo u medijima za oglašavanje da pročitamo neku novu vijest o tome kako su „hakeri“ (što je netačno korišten termin) provalili u neku kompaniju, kako su izašli novi podaci o špijuniranju provedenom od strane NSA ili kako je nekoliko desetina hiljada korisničkih podataka neke kompanije završilo javno dostupno na internetu.

Kompanije širom svijeta se sada bore da dokažu ne samo da su njihovi proizvodi najjeftiniji i najpouzdaniji, nego i najsigurniji kada je u pitanju privatnost njihovih korisnika. Iako je razvoj desktop aplikacija u današnje vrijeme u velikom opadanju u odnosu na web aplikacije i mobilne aplikacije, i dalje se desktop aplikacije smatraju najsigurnijima zbog činjenice da u većini njih korisnički podaci ostaju tamo gdje su i generisani – na korisničkom računaru.

Međutim, *Microsoft*-ovi alati se zbog njihove zatvorenosti jako rijetko smatraju sigurnima i u vrlo malom broju slučaja se koriste u poslovnom svijetu. Ono što je mene zanimalo jeste da li se može napraviti sigurna aplikacija koja će da vrši enkripciju i heširanje unesenih podataka koristeći *Microsoft*-ov vlastiti programski jezik i njegovu standardnu biblioteku dizajniranu za kriptografiju (koja se u ovom slučaju naziva *System.Security.Cryptography*).

Dakle, koristeći *Visual Basic .NET* i *System.Security.*Cryptography biblioteku pokušaću da implementujem različite kriptografske algoritme i upoređivaću rezultate implementacije kao i samu lakoću implementiranja sa ostalim, mnogo popularnijim programskim jezicima kada je u pitanju rad sa kriptografijom da bi došao do zaključka da li je *Microsoft* dorastao zadatku. Odlučio sam da implementujem sledjeće algoritme (uz napomenu da prva tri algoritma služe za enkripciju podataka, dok druga tri služe za hešovanje):

1. DES
2. Triple DES
3. AES
4. MD5
5. SHA-1
6. SHA-2

**Ključne riječi:** *kriptografija, enkripcija, hešovanje, Visual Basic, .NET, implementacija*

# Enkripcija podataka

## Uvod u enkripciju podataka

Da bi se podaci zaštitili, odavno je prestalo da bude dovoljno da se svaki karakter zamijeni nekim drugim i da tako budemo sigurni da niko (osim nas) neće moći da dođe do naših originalnih podataka (nazvanih *plaintext* u kriptografskom žargonu) isključivo gledajući u izmjenjeno stanje u kojem se nalaze podaci nakon što smo svaki karakter zamijenili nekim drugim (stanje koje se u kriptografskom žargonu naziva *cyphertext*). Napretkom tehnologije, napredovali su i načini kako otkriti originalne podatke, tako da su algoritmi za sakrivanje podataka morali da postaju sve komplikovaniji i komplikovaniji.

Agencija koja je zadužena za standardizaciju kriptografskih algoritama je američki Institut standarda i tehnologija[[1]](#footnote-1) (eng. *National Institute of Standards and Technology* ili skraćeno NIST). Kada jedan algoritam bude „probijen“ (u smislu da je pronađen dovoljno efikasan način da se zaobiđe), NIST objavljuje konkurs za novi algoritam koji će da zamijeni njegovo mjesto. Svi potom prijavljeni algoritmi prolaze kroz žestoke kontrole te se na kraju bira samo jedan koji će institut da prihvati kao standard.

Kao prvi ozbiljan standard za koji se pomišljalo da se neće moći probiti decenijama, izabran je *Data Encryption Algorithm* (skraćeno DEA). Ovaj algoritam je prvi put objavljen 1977. godine, a kao službeni standard je prihvaćen dvije godine nakon objave, nakon čega je preimenovan u *Data Encryption Standard* (skraćeno DES). Međutim, ovaj standard se nije zadržao onoliko koliko je planirano već je napretkom računara uspješno razbijen. Prvo je 1992. Objavljen teoretski način probijanja algoritama, zatim je 1997. algoritam prvi put uspješno probijen. Već 1998. Je napravljena mašina koja je taj posao skratila na samo 56 sati, a već pola godine nakon toga je posao skraćen na manje od dan (tačnije 22 sata i 15 minuta), čime je dokazano da je vrijeme da se pokrene potraga za novim algoritmom koji će da postane standard.[[2]](#footnote-2)

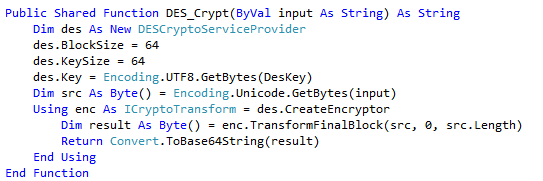
Kao najbrža zamjenu koju su uspjeli pronaći za DES algoritam se pojavio Triple DES 1998. godine, algoritam koji u osnovi ponavlja proces enkriptovanja DES algoritmom tri puta čime se izbjegavaju greške iskorištene u DES algoritmu. Pošto je postupak enkriptovanja isti kao i sa DES algoritmom, nije postojalo nekompatibilnosti sa uređajima koji su podržavali DES standard. Međutim, zbog svog dizajna koji podrazumijeva trostruku enkripciju DES algoritmom, Triple DES je zbog svoje nedovoljno velike brzine u izvršavanju bio i ostao samo privremena zamjena dok se nije uspostavio novi standard, iako za Triple DES ni danas ne postoji efektivan napad.

Kao posljednji i još uvijek aktuelan algoritam koji se koristi za enkripciju se pojavio algoritam pod nazivom Rijndael koji je od strane NIST-a prihvaćen kao standard 2002. godine nakon temeljite dugogodišnje provjere njegovog kvaliteta. Kada je prihvaćen kao standard, algoritam je dobio novo ime pod kojim je i danas poznat: *Advanced Encryption Standard* (skraćeno AES). I dalje se smatra iznimno sigurnim i postoje samo neki teoretski napadi na najslabiju verziju AES-a koja koristi 128-bitni ključ za enkripciju.

Sva tri opisana standarda u ovom poglavlju su vrlo bitna za istoriju zaštite podataka i kao takve sam imao dužnost da ih pokušam implementirati u *Visual Basic*-u i *.NET* *framework*-u koristeći standardnu *System.Security.Cryptography* biblioteku.

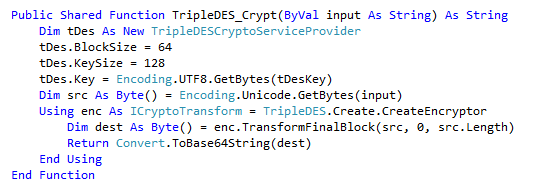
## 2.2. Implementacija DES algoritma

DES kao najstarija vrsta enkripcije je davno razbijena, ali je vrlo bitna za istoriju enkripcije. Zato sam odlučio da ga implementiram u svoj rad. Važno je napomenuti da se IV daje nasumično tako da će rezultat funkcije svaki put biti drugačiji. Ključ je konstantan i definisan je u varijabli DesKey i njegova vrijednost je "ABCDEFGH". DES koristi blokove od 65 bita i isti toliki ključ. Koristi se *DESCryptoServiceProvider* za enkripciju.



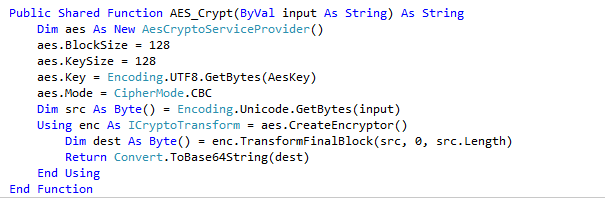
## 2.3. Implementacija TripleDES algoritma

TripleDES je nasljednik DES algoritma koji radi na istom principu kao i DES algoritam, samo se čisti tekst provlači kroz DES algoritam tri puta. To dosta usporava proces enkripcije te zbog toga nije uveliko u upotrebi.



## 2.4. Implementacija AES algoritma

AES je trenutno najrasprostranjeniji algoritam za enkricpiju. Odlučio sam da implementiram 128-bitnu verziju. Ključ je konstantan i glasi: "ABCDEFGHIJKLMNOP". IV je ponovo automatski generisan tako da će rezultat stalno da bude drugačiji.



# Implementacija hash algoritama

## 3.1. Uvod u proces hash-ovanja

Napraviti *hash* on nekog teksta znači provući ga kroz matematički algoritam koji zadovoljava sledjeća dva uslova:

1. Tekst bilo koje dužine kao *input* treba da vrati tekst fiksne dužine kao *output*.
2. Dva različita teksta ne smiju da proizvedu isti *output*.

Hash algoritmi se najviše koriste u za provjeru da li su dvije datoteke jednake. Ovo je vrlo korisno u slučaju da šaljemo konfedencijalne datoteke preko interneta jer na vrlo lak način možemo da provjerimo da li je neko te datoteke izmijenio na putu od polazišta do odredišta. Korišnjenjem *hash*-eva takođe možemo da provjerimo i da li smo na odredištu dobili kompletnu datoteku ili se prenos u nekom trenutku prekinuo. Provjera je vrlo jednostavna. Sve što trebamo da uradimo jeste da izračunamo *hash* na polazištu i na odredištu korišćenjem istog algoritma te da usporedimo njihove vrijednosti. Ukoliko se vrijednosti poklapaju, znači da je datoteka uspješno prenijeta i da na oba računara imamo potpuno identičnu datoteku.

Postoji nekoliko različitih algoritama koji nam to omogućavaju, od kojih su svakako najznačajniji MD5 i SHA algoritmi. MD5 algoritam je konstantno u upotrebi od 1992. godine. Iako postoji nekoliko različitih algoritama napravljenih od istih autora (Ralph Merkle i Ivan Damgård) jedino je MD5 doživio svjetsku slavu i postao korišten širom svijeta (pored njega, postoje još i MD2, MD4 i MD6 algoritmi). MD5 proizvodi *output* dugačak 128 bita. Međutim, otkriveni su mu propusti i otkriveno je da dvije različite poruke mogu da proizvedu jednak *output*, što implicira da dolazi do kolizije. Ti propusti su sve više iskorištavani sve do 2013. godine kada su Xie Tao, Fanbao Liu i Dengguo Feng uspjeli da implementiraju propuste tako da se kolizije mogu pronaći u manje od jedne sekunde na prosječnom računaru[[3]](#footnote-3).

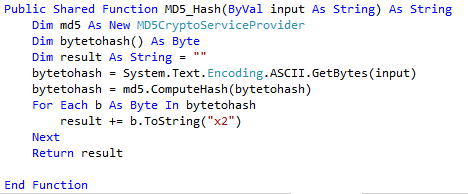
Postalo je jasno da je MD5 algoritam nesiguran i nastala je potreba da se pronađe algoritam koji će da ga uspješno zamijeni. Tu je opet NIST došao na scenu koji je prihvatio SHA-1 algoritam (gdje je SHA skraćeno od *secure hash algorithm*) kao standard. SHA-1 je razvila američka Agencija za nacionalnu sigurnost (eng. *National Security Agency*). Napretkom tehnologije 2005. godine pronađene su mane i u ovom algoritmu. Iako do sada nisu pronađene nikakve kolizije, postala je jasna potreba za novim algoritmom i tada stupa na scenu SHA-2.

SHA-2 je postao dostupan za javnost 2001. godine i njegov sam algoritam se dosta razlikuje u odnosu na njegovog prethodnika. Može da proizvodi *output* u nekoliko različitih varijanti: 224 bita, 256 bita, 384 bita i 512 bita. Trenutno se smatra sigurnim i u širokoj je upotrebi, međutim već se traži njegov nasljednik SHA-3 algoritam.

Odlučio sam da implementujem sva tri najpoznatija *hash* algoritma. Postoje već predefinisani algoritmi u *Visual Basic .NET* okruženju i sve što je potrebno jeste pravilno implementovati funkcije iz *System.Security.Cryptography* biblioteke.

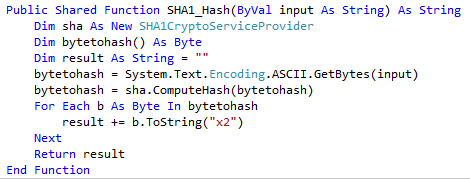
## 3.2. Implementacija MD5 algoritma

Implementaciju MD5 algoritma vršim u funkciji *MD5\_Hash* koja se izvršava kada je na formi označena pravilna RadioButton kontrola. Koristio sam *MD5CryptoServiceProvider* funkciju za izračunavanje *hash* vrijednosti. Pošto ova funkcija prima vrijednosti u bajtovima, potrebno je da *input* pretvorimo u vrijednost u bajtovima. Za to je korištena *bytetohash* varijabla. Konačan rezultat se sprema u varijablu *result*, a vrijednost varijable *result* se vraća nazad u glavni program kao rezultat funkcije.



## 3.3. Implementacija SHA-1 algoritma

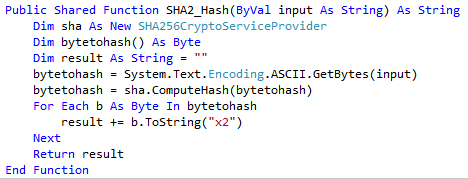
Implementaciju SHA-1 algoritma vršim u funkciji *SHA1\_Hash* koja se izvršava kada je na formi označena pravilna RadioButton kontrola. Koristio sam *SHA1CryptoServiceProvider* funkciju za izračunavanje *hash* vrijednosti. Pošto ova funkcija prima vrijednosti u bajtovima, potrebno je da *input* pretvorimo u vrijednost u bajtovima. Za to je korištena *bytetohash* varijabla. Konačan rezultat se sprema u varijablu *result*, a vrijednost varijable *result* se vraća nazad u glavni program kao rezultat funkcije.



## 3.4. Implementacija SHA-256 algoritma

SHA-256 je naziv za SHA-2 algoritam koji proizvodi *output* u dužini od 256 bita.

Implementaciju SHA-2 algoritma vršim u funkciji *SHA2\_Hash* koja se izvršava kada je na formi označena pravilna RadioButton kontrola. Koristio sam *SHA256CryptoServiceProvider* funkciju za izračunavanje *hash* vrijednosti. Pošto ova funkcija prima vrijednosti u bajtovima, potrebno je da *input* pretvorimo u vrijednost u bajtovima. Za to je korištena *bytetohash* varijabla. Konačan rezultat se sprema u varijablu *result*, a vrijednost varijable *result* se vraća nazad u glavni program kao rezultat funkcije.



# Zaključak

Iako se Visual Basic kao i ostatak Microsoft-ovih tehnologija slabo koriste za pravljenje programa preko kojih se upravlja delikatnim podacima (samo iz jednog razloga: sve je zatvoren kod), sami algoritmi su vrlo dobro implementirani. Sve funkcije se mogu pronaći u biblioteci koja dolazi uz sam Visual Studio (*System.Security*) i na programerima jeste jedino posao da ih kvalitetno implementuju u svoj program. Smatram da sam za primjer korištenja enkripcijskih i *hash* funkcija odradio dobar posao. Smatram da je potrebno da svaki mladi programer istraži ovu biblioteku u *Visual Studiju*  i smatram da je na nama budućim programerima da proširimo svijest o tome koliko je potrebno da se maksimalno koristi enkripcija u budućim softverskim rješenjima, pošto u skorije vrijeme očekujemo da se mnogo novih uređaja konektuje na internet, od kojih neka softverska rješenja mogu i odlučiti o nečijem životu (tipa medicinska pomagala i pametni auti).

1. <http://nist.gov/> [↑](#footnote-ref-1)
2. DES hronologija: <https://en.wikipedia.org/wiki/Data_Encryption_Standard#Chronology> [↑](#footnote-ref-2)
3. Izvor: https://eprint.iacr.org/2013/170.pdf [↑](#footnote-ref-3)