UDC: 004.056.55 Info M str. 4-17

IMPLEMENTACIJA STEGANOGRAFIJE I KRIPTOGRAFSKIH ALGORITAMA U MATLAB-U
IMPLEMENTATION OF STEGANOGRAPHY AND CRYPTOGRAPHIC ALGORITHMS IN MATLAB

Boris Malčić, prof. dr Zoran Đurić, prof. dr Slavko Šajić

REZIME: U ovom radu je prvo dat opis i značaj steganografije, kao i osnovne informacije oko standardnih algoritama enkripcije kao što su DES, AES i RSA. Takođe, u uvodnom dijelu su opisani osnovni pokazatelji kvaliteta primjenjene steganografije kao što su MSE, PSNR i NCC. Nadalje, veliki značaj ovog rada, osim što smo implementirali aplikaciju u MATLAB-u za video steganografiju sa i bez DES i AES enkripcije/dekripcije, ogleda se u tome što smo prikazali i sve izvorne kdove navedenih implementiranih funkcija u MATLAB-u koji čitaocu omogućavaju da samostalno ponovi isti postupak i provjeri dobijene rezultate, **što može biti** od posebnog značaja u edukativnim svrhama kako bi se spojila teorija i praksa. U sljedećem dijelu našeg rada prikazujemo izračunate vrijednosti već pomenutih pokazatelja kvaliteta u slučaju primjene implementirane *LSB* video-steganografije, kao i poređenja vremena izvršavanja pojedinih steganografskih opcija sa primjenom enkripcije/dekripcije i digitalnog potpisivanja primjenom RSA. Na kraju rada, kao zaključak ističemo najvažnija zapažanja i navodimo smjernice za eventualna buduća istraživanja.

KLJUČNE REČI: AES, kriptografija, DES, LSB steganografija, *MSE*, *NCC*, *PSNR*, RSA.

ABSTRACT: In this paper, a description and importance of steganography is first given, as well as basic information about standard encryption algorithms such as DES, AES and RSA. Also, the introductory part describes the basic quality indicators of applied steganography such as *MSE*, *PSNR* and *NCC*. Furthermore, the great importance of this work, apart from the fact that we implemented an application in MATLAB for video steganography with and without DES or AES encryption/decryption, importance is also reflected in the fact that we have presented all the source codes of the implemented functions in MATLAB that allow the reader to repeat the same procedure himself and check the results obtained. All of this is of particular importance for educational and further research purposes in order to combine theory and practice. In the next part of the paper, we present the calculated values of the already mentioned quality indicators in the case of the application of implemented LSB video-steganography, as well as the comparison of the execution time for individual steganographic options with the application of encryption/decryption and digital signature using RSA. At the end of the paper, as a conclusion, we highlight the most important observations and indicate guidelines for possible future research.

KEY WORDS: AES, cryptography, DES, LSB steganography, *MSE*, *NCC*, *PSNR*, RSA.

1. UVOD

Poruka otvorenog ili izvornog teksta (*eng.*plaintext message) može biti skrivena na jedan od tri načina: primjenom steganografije, primjenom kriptografije ili njihovom kombinacijom. Pod pojmom steganografije se misli na umjetnost prenosa podataka pomoću skrivanja izvornih podataka koje želimo prenijeti pomoću nekih drugih (većih) tekstualnih ili multimedijalnih datoteka (npr. slika, video ili audio zapisa) [1]. Konkretno, u ovom radu je objašnjen postupak prenosa audio zapisa pomoću video zapisa primjenom steganografije čija implementacija je izvršena u MATLAB-u. Naravno, kao izvorna poruka može se koristiti i neki video zapis, ali u tom slučaju kao prenosni medijum bi nam trebao video zapis mnogo veće veličine kao što je npr. realizovano u [2]. Metode steganografije prikrivaju postojanje poruke, dok metode kriptografije čine poruku nerazumljivom nelegalnim učesnicima u komunikaciji korišćenjem raznih transformacija nad izvornom porukom. Najčešće je izvorna poruka u vidu nekog tajnog teksta čiji sadržaj treba zaštiti od nepoželjnih presretača komunikacije prilikom prenosa od predajnika (osobe ili uređaja koji generiše poruku za slanje) do prijemnika (onog kome je zaista poruka namijenjena). Izvorna poruka može biti u bilo kojem obliku (formatu) kao što su npr. slike, audio i video zapisi. Jednostavna forma steganografije, ali za koju je potrebno mnogo vremena, jeste forma u kojoj neki raspored riječi ili slova, unutar naizgled otvorenog teksta, izražava skrivenu poruku. Na primjer, redoslijed prvih slova svake riječi ukupne poruke može da sačinjava skrivenu poruku. U ovom radu, predstavljena je mplementacija steganografije u MATLAB-u koja izvorni audio zapis skriva u video zapis, tzv. video steganografija. Što se tiče područja skrivanja podataka, video steganografija ima veću prednost u poređenju sa drugim steganografskim tehnikama budući da koristi video (tj. ima znatno veću redundansu za skrivanje izvorne informacije) kao prenosni medij [3].

Steganografija ima brojne nedostatke u poređenju sa enkripcijom. Naime, straganografija zahtijeva korišćenje mnogo većih zaglavlja u paketima podataka koji se prenose u cilju sakrivanja relativno malog broja (nekoliko) bitova izvorne (tajne) informacije. Nevidljivost izvorne poruke, nosivost prenosnog medijuma i sigurnost u smislu vršnog odnosa signala i šuma (*eng. **P**eak **S**ignal to **N**oise **R**atio* - ***PSNR***) i robusnosti su ključni izazovi steganografije [4]. Statistička distorzija između originalnih (nepromijenjenih) slika i stego-slika se mjeri korištenjem srednje-kvadratne greške (*eng. **M**ean **S**quare **E**rror* - ***MSE***) i već pomenutog *PSNR*, dok se stepen sličnosti između slika procjenjuje korištenjem normalizirane unakrsne korelacije (*eng. **N**ormalized **C**ross **C**orrelation* - ***NCC***). Takođe, kada se otkrije sistem na kome je implementirana steganografija (npr. steganografija bazirana na bitu najmanje težine, *eng.* **L**east **S**ignificant **Bi**t**s** – **LSBs**, u bajtovima nekog frejma slike, videa itd.), onda čitav implementirani sistem steganografije postaje praktično bezvrijedan. Postoje različite steganografske tehnike za skrivanje informacija. Jedna od njih je prethodno spomenuta LSB metoda. Može se konstantovati da su osnovni nedostaci LSB metode: 1. serijski odabir piksela

unutar video okvira (*eng.* frame) koji se koristi za utiskivanje informacija u taj okvir, a samim tim tu je prisutna i 2. slaba imunost na elektronske smetnje/napade. Iz tih razloga, u radu [5] je predložen algoritam tzv. *viteške turneje* za slučajni odabir piksela, kao i metoda šifriranja ključne funkcije koja se koristi za šifriranje tajne poruke kako bi se povećala sigurnost i robusnost. Nadalje, problem serijskog odabira piksela unutar video okvira, može se prevazići ako način utiskivanja informacionih bita izvorne poruke zavisi od neke vrste ključa, ili ako se prije korišćenja steganografije uradi još i enkripcija (npr. zbog veće brzine enkripcije mogao bi se iskoristiti neki od simetričnih blok šifarskih sistema – tj. algoritama sa tajnim ključem). Blokovski šifarski algoritam je šema šifrovanja/dešifrovanja u kojoj se jedan blok otvorenog teksta procesira kao jedna cjelina i dalje se koristi za dobijanje jednog bloka šifriranog teksta (tzv. šifrata - *eng.* cipher) najčešće jednake dužine kao što je bila dužina ulaznog bloka otvorenog teksta.

DES standard šifrovanja podataka (*eng.* **D**ata **E**ncryption **S**tandard - **DES**) je bio najčešće korišten simetrični kripotografski algoritam do kraja 90-tih kada je po prvi put (1997. godine) razbijen čime je narušena njegova sigurnost, a od maja 2005. godine je povučen iz upotrebe [6]. Tačnije, od tada NIST (*eng.* **N**ational **I**nstitute of **S**tandards and **T**echnology at the U.S. Department of Commerce – **NIST**) ne preporučuje korišćenje DES kriptografije zbog njegove relativno male dužine ključa (koja iznosi 56 bita) čime je omogućeno relativno brzo razbijanje šifre pomoću današnjih računara. DES koristi 64-bitni ulazni blok podataka i 56-bitni ključ za enkripciju (isti ključ se koristi i za dekripciju) i baziran je na Feistel-ovoj strukturi. Mnogi blokovski kriptografski algoritmi imaju Feistel-ovu strukturu. Takva struktura se sastoji od određenog (npr. često 10, 12, 14, 16 ili više) broja identičnih postupaka obrade (često se ovi identični postupci obrade nazivaju rundama - eng. **rounds**) nad ulaznim blokom podataka. U opštem slučaju, što je veći broj rundi obrade – posmatrani kriptografski algoritam će biti otporniji na napade. U svakoj rundi najčešće se vrši zamjena na jednoj polovini podataka koji se obrađuju, nakon čega slijedi permutacija koja zamjenjuje te dvije polovine ulaznog bloka podataka (npr. u nastavku teksta dvije polovine po 32 bita koji sačinjavaju 64-bitni blok kod DES enkripcije/dekripcije su označene kao ***i***-ti desni 32-bitni podblok – $\boldsymbol{R_i}$ i ***i***-ti lijevi 32-bitni podblok – $\boldsymbol{L_i}$ od ukupne ***i***-te blokovske 64-bitne poruke koja se može predstaviti konkatenacijom kao $\boldsymbol{R_iL_i}$). Originalni ključ je često proširen (*eng.* expanded) tako da se za svaku rundu koristi poseban podključ runde koji se poznatim i tačno određenim postupcima dobija iz originalnog ključa.

Dvije važne metode kriptoanalize su diferencijalna kriptoanaliza i linearna kriptoanaliza. Pokazalo se da je DES bio vrlo otporan na ove dvije vrste napada, ali praktično nijedan od standardizovanih algoritama kriptografije ne može biti imun na "prosti napad silom" (*eng.* brute-force attack) gdje se redom provjeravaju sve moguće kombinacije ključeva dok se ne dešifruje šifrat, a u tu svrhu se mogu raditi i distribuirani napadi usaglašeni sa više radnih mašina (PC računara, radnih stanica, laptopa, servera, itd.). Naravno, kod modernih algoritma je teško (praktično nemoguće) razbiti šifru u realnom vremenu. Takođe, iako su brojni simetrični kriptografski šifrati razvijeni i predloženi od davne 1977. godine kada je zvanično uveden DES, treba istaći da DES ostaje i dalje najvažniji simetrični algoritam za kriptografiju. Naime, ako se detaljno prostudira i razumije princip rada DES-a, onda je mnogo lakše razumjeti i novije/savremene simetrične kriptografske algoritme. Većina algoritama simetrične blok enkripcije u trenutnoj upotrebi zasniva se na strukturi koja se naziva Feistel-ova blok šifra. Feistel je davno predložio da se aproksimira idealni blok šifrat korištenjem koncepta proizvoda šifrata. Ustvari Feistel-ova blok šifrat vrši dvije ili više jednostavnih operacija nad podatacima u nizu na takav način da konačni rezultat (šifrovani podatak) ili proizvod bude kriptografski robusniji od bilo koje od komponentnih šifrata (šifriranih podataka u rundama). Suština pristupa je da se razvije blok šifrat sa dužinom ključa od k bita i dužinom bloka od n bita, dozvoljavajući ukupno 2^k mogućih transformacija, umjesto 2^n! transformacija dostupnih kod idealnog blok šifrata. Konkretno, Horst Feistel[1] je predložio upotrebu algoritma koji naizmjenično vrši zamjene i permutacije. Proces dešifrovanja pomoću Feistel-ovog šifrata je u suštini isti kao i proces šifrovanja [7]. Naime, kod DES dekripcije potrebno je u suštini primjeniti isti postupak kao kod enkripcije ali u inverznom redoslijedu (šematski posmatrano odozdo prema gore) tako da će se u prvoj iteraciji primijeniti podključ K_{16}, pa u drugoj iteraciji podključ K_{15}, ..., a tek u 16. iteraciji podključ K_1. Matematički, taj algoritam dekripcije možemo iskazati sa (1) i (2), odnosno sljedećim formulama:

$$R_{n-1} = L_n \tag{1}$$

$$L_{n-1} = R_n + f(L_n, K_n), \tag{2}$$

gdje je znakom označena operacija sabiranja po modulu 2, a $\boldsymbol{R_{16}L_{16}}$ je permutovani ulazni blok za DES dešifrovanje i $\boldsymbol{R_0L_0}$ je predizlazni blok dešifrovane poruke. Primjer implementacije u MATLAB-u postupka DES dekripcije prikazan je u trećem dijelu rada sa listingom 3.13.

Kada se shvati princip rada DES algoritma, onda je lako implementirati i tzv. trostruki DES (*eng.* **T**riple **D**ata **E**ncryption **A**lgorithm – **TDEA** ili 3DES) kod koga se postupak enkripcije/dekripcije svodi na tri ulančane primjene DES algoritma sa „najčešće" različitim ključevima K_1, K_2 *i* K_3 primjenjenim kod pojedinačnih DES algoritama u lancu - kao što je prikazano na slici 1. Na istoj slici je isprekidanom linijom naznačeno da je izlazni blok podataka iz TDEA enkripcije (označen sa ***O)*** ujedno ulazni blok podataka u procesu dekripcije (na slici 1 označen sa ***I)***. Nadalje, neka $E_K(I)$ i $D_K(I)$ predstavljaju DES enkripciju (*eng.* ciphering) i dešifrovanje/dekripciju (*eng.* deciphering) 64-bitnog bloka podataka I pomoću DES algoritma primjenom ključa K, respektivno. Konačno, TDEA je detaljno opisan u ANSI X9.52, a složene operacije TDEA enkripcije i dekripcije su:

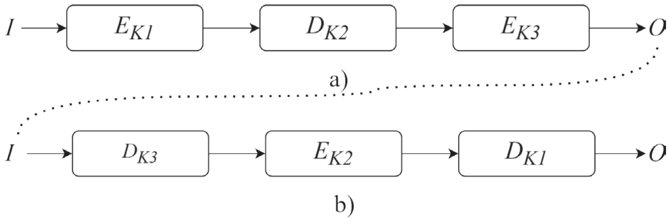
1. Operacija TDEA enkripcije je transformacija 64-bitnog bloka I u 64-bitni blok O na sljedeći način

$$O = E_{K3}(D_{K2}(E_{K1}(I))),$$

1 Horst Feistel (1915 – 1990) je bio njemačko-američki kriptograf i fizičar koji je radio za IBM i MIT, a njegov istraživački rad u sklopu IBM-a je imao veliki značaj za realizaciju Lucifer i DES šifrata.

2. TDEA operacija dekripcije je transformacija 64-bitnog bloka *I* u 64-bitni blok *O* primjenom

$$O = D_{K1}(E_{K2}(D_{K3}(I))).$$



Slika 1: a) Blok šema postupka TDEA enkripcije; b) Blok šema postupka TDEA dekripcije

Nadalje, napredni standard šifrovanja (*eng.* **A**dvanced **E**ncryption **S**tandard - **AES**) je blok šifarski algoritam prvobitno namijenjen da zamijeni DES za komercijalne aplikacije. AES koristi 128-bitnu veličinu *plaintext* ulaznog bloka podataka i veličinu ključa od: 128, 192 ili 256 bita. Bitno je napomenuti da AES ne koristi Feistel-ovu strukturu. Umjesto toga, svaka runda enkripcije se sastoji od četiri zasebne funkcije: zamjena bajtova, permutacija, aritmetičke operacije nad konačnim poljem, i XOR sa ključem. Nacionalni institut za standarde i tehnologiju SAD-a, tj. NIST, objavio je AES kao standard 26. novembra 2001. godine u publikaciji **FIPS PUB – 197** (*eng.* **F**ederal **I**nformation **P**rocessing **S**tandards **PUB**lication – **FIPS PUB**) [7]. Naime, NIST je odabrao **Rijndael**-ov algoritam kao predloženi AES algoritam. Rijndael-ov prijedlog za AES definisao je šifrat u kojem se dužina bloka ulaznog podatka i dužina ključa može nezavisno izabrati na jednu od vrijednosti: 128, 192 ili 256 bita. AES specifikacija koristi sve tri veličine ključa alternativno, ali ograničava dužinu bloka *plaintext*-a na 128 bita.

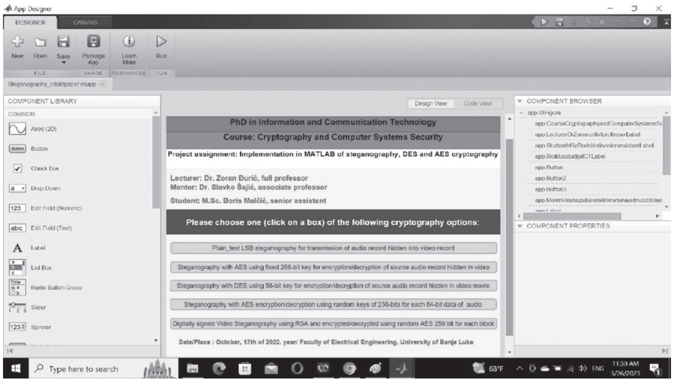
U nastavku rada slijedi opis implementacije *Steganography.mlapp* aplikacije u MATLAB-u. Potom u trećem dijelu ovog rada prikazani su izvorni kodovi implementiranih funkcija u MATLAB-u, a dat je i njihov opis. U četvrtoj sekciji članka predstavljeni su rezultati simulacije i dobijene vrijednosti za *MSE, PSNR* i *NCC* kod implementirane *LSB* video-steganografije. Konačno, u petom dijelu rada, date su preporuke za buduća istraživanja i na kraju je prikazana osnovna korištena literatura.

2. PRAKTIČNA IMPLEMENTACIJA

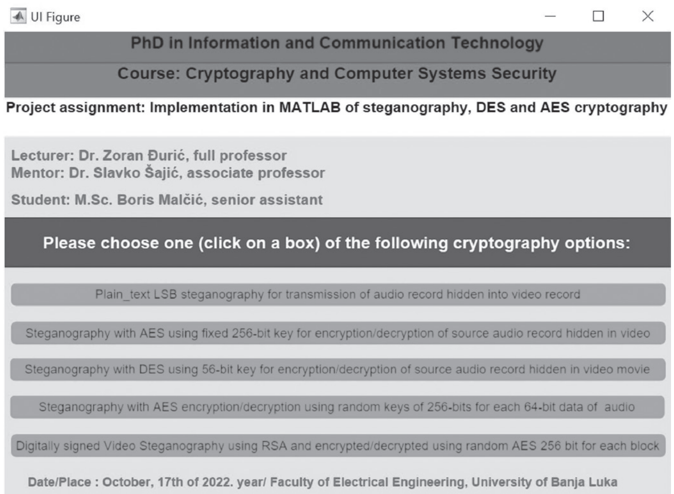
Konačni rezultat implementirane steganografije, te njene kombinacije sa DES i AES baziranom kriptografijom (konkretno, enkripcijom i dekripcijom izvornog audio zapisa) u MATLAB-u jeste grafičko-korisničko programsko okruženje (GUI *Steganography.mlapp* aplikacija prikazana na slici 2) koje se dizajnira i pokreće u komandnom prozoru MATLAB-a pomoću komande:

```
>> appdesigner.
```

Konkretno, u nastavku je korišćena **MATLAB R2016b** verzija instalirana na **Windows 10 Pro** operativnom sistemu HP laptopa sa **64**-bitnim procesorom **AMD Ryzen 3 3200U with Radeon Vega Mobile Gfx 2.60 GHz i 8GB** instalirane **RAM** memorije od koje je **5.95 GB RAM** memorije dostupno za korisničke programe.



Slika 2: Radni prozor **App Designer** okruženja



Slika 3: GUI implementirane aplikacije ***Steganography.mlapp***

U nastavku, navodimo redom izvorne kodove i objašnjenje većine implementiranih funkcija u ***Steganography.mlapp***.

3. IMPLEMENTACIJE KORIŠTENIH FUNKCIJA U MATLAB-U

Listinzi implementiranih funkcija su:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% Izvorni kod za prvo zeleno dugme sa slike 3 koje poziva funkciju:    %%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
plain_text_steganography (app, event) % Funkcija u 1. zelenom dugmetu GUI app sa slike 3
%% Autori: M.Sc. Boris Malčić, prof. dr Zoran Đurić, prof. dr Slavko Šajić
%        Elektrotehnički fakultet, Univerzitet u Banjoj Luci (ETFBL, UNIBL)
%        Republika Srpska/Bosna i Hercegovina (RS/BiH)
%Sva prava zadržavaju autori (kontakt e-mail: boris.malcic@etf.unibl.org) © copyright reserved.
% Autori se odriču bilo kakve odgovornosti ako budete koristili bilo koju od njihovih funkcija!
clear all; % Prvo, obrišemo sve prethodno korišćene varijable u MATLAB-ovom komandnom
% prozoru
close all; % Potom, zatvorimo sve prethodno korištene slike u MATLAB-u.
clc;  % Obrišemo komandni prozor od prethodnih komandi.
tic; % Startujemo vremenski brojač kako bi mjerili vrijeme koje je potrebno da se izvrše sve
% donje funkcije do posljednje komande toc
[y,Fs] = audioread('mySpeech.wav'); % Učitamo u MATLAB mySpeech.wav zapis koji će
% služiti kao "plaintext"poruka
```

```
sound(y,Fs) % Potom, poslušamo ulazni plaintext (.wavzapis) koji će biti skriven u video zapisu % koristeći steganografiju
wavebinary = dec2bin(typecast(single(y), 'uint8'), 8 ) - '0';% Konvertujemo plaintext poruku
% u binarni format
vidObj = VideoReader('Superman.mp4');% Učitamo u MATLAB video zapis koji ćemo
% koristiti za steganografiju
vidHeight = vidObj.Height; % Potom, očitamo veličinu (visinu i širinu frejmova) od video
% zapisa tj. Superman.mp4 video zapisa
vidWidth = vidObj.Width;
mov = struct('cdata',zeros(vidHeight,vidWidth,3,'uint8'), 'colormap',[]); % Kreiramo
% instancu strukture video zapisa (tj. eng.mov – movie)
k = 1; % Krećemo od 1. frejma i donjom komandom prolazimo kroz sve frejmove
while hasFrame(vidObj) % Dok god ima frejmova u video objektu vidObj (tj. Superman.mp4)
mov(k).cdata = readFrame(vidObj); % Kopiramo frejm od vidObj u k-ti frejm strukture mov
k = k+1; % Prelazimo u sljedeći frejm
end
% Potom, dolje pozivamo funkciju koja implementira našu steganografiju
OutputVideo = NasaSteganografija(wavebinary, mov);
snimi_u_avi_format_video(OutputVideo);% izlazni video snimimo u stegoSuperman.avi
% Nadalje, na prijemnoj strani, trebamo iz stego video zapisa (tj. OutputVideo) izvući audio zapis
RXvidObj = VideoReader('stegoSuperman.avi'); % Učitamo video zapis stegoSuperman.avi u koji je utisnut audio zapis mySpeech.wav
vidHeight = RXvidObj.Height; % Očitamo veličinu frejmova od stegoSuperman.avi zapisa, tj. % visinu i širinu frejmova u pikselima
vidWidth = RXvidObj.Width;% širina frejma u pikselima
r_mov = struct('cdata',zeros(vidHeight,vidWidth,3,'uint8'), 'colormap',[]); % Formiramo
% video strukturu (r_mov) u koju ćemo učitati primljeni video zapis (stegoSuperman.avi)
k = 1; % Polazimo od 1. frejma
while hasFrame(RXvidObj) % dok god ima frejmova u primljenom videu RXvidObj (tj. u
% stegoSuperman.avi)
r_mov(k).cdata = readFrame(RXvidObj); % Kopiramo frejm iz RXvidObj kao k-ti frejm u
% r_mov(k)
k = k+1; % Prolazimo kroz sljedeći frejm
end
[y_format,Fs] = audioread('myFormatSpeech.wav'); % Učitamo govorni signal koji služi kao
% format – tj. unaprijed se zna u kojem format predajnik vrši steganografiju
sound(y_format,Fs) % Poslušamo formatirani audio zapis. Naime, smatramo da su se predajnik i
% prijemnik unaprijed usaglasili oko formata audio zapisa koji će se koristiti u videu primjenom
% steganografije.
% Dolje, pomoću primljenog steganografskog video zapisa i formata govornog audio zapisa
% izdvajamo govorni audio zapis - wavdata
[wavdata] = InverzSteganografija(r_mov, y_format ); % Primljeni audio zapis
sound(wavdata,Fs); % Poslušamo primljeni audio zapis (wavdata) koji bi trebao biti isti kao
% poslati audio zapis - mySpeech.wav
toc; % Zaustavimo vremenski brojač koji će nam poslije toga u komandnom prozoru MATLAB-a
% prikazati u sekundama ukupno trajanje izvršavanja svih funkcija od tic do toc komande.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Listing 3.1: Izvorni kod funkcija koje aktivira prvo zeleno dugme naziva ***Plain_text LSB steganography for transmission of audio record hidden into video record*** sa slike 3

U listingu 3.1 su **bold**-ovanim slovima napisane komande, a poslije znaka % navedeni su komentari, tj. opis šta smo implementirali pojedinim komandama. Redom, kada proučavate funkcije u ovom listingu imate neke od ugrađenih MATLAB-ovih funkcija čiji opis možete vidjeti u dokumentaciji instaliranog MATLAB-a ili ako ukucate naredbu ***help*** *naziv_funkcije* u komandni prozor MATLAB-a. Prva naša funkcija, čiju implementaciju i opis trebamo objasniti, prikazana je listingom 3.2.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function OutputVideo = NasaSteganografija(wavebinary, mov)
%% Autori: Boris Malčić, Zoran Đurić, Slavko Šajić
% Elektrotehnički fakultet, Univerzitet u Banjoj Luci, Republika Srpska, Bosna i Hercegovina
% Sva prava (© copyright) zadržavaju autori i nije dozvoljeno korištenje bez referenciranja članka
% Svrha: Za edukativne svrhe implementacija steganografije utiskivanjem audio zapisa u video
% zapis.
% Opis zadatka: Implementacija steganografije primjenom LSB (eng.Least Significant Bit)
% Cilj: Sakriti audio zapis (.wav formata) u video zapis (.avi formata)
% Ulazni argumenti funkcije: - mov je ulazni video zapis u koji skrivamo audio zapis
% - wavbinary je binarni zapis audio snimka formata .wav
% Za snimanje audio zapisa pomoću mikrofona smoiskoristili MATLAB-ov primjer iz
% dokumentacije pod nazivomRead from Microphone and Write to Audio File.
% Izlaz Read from Microphone and Write to Audio File.mfunkcije je mySpeech.wav audio zapis
% koji treba „sakriti u video zapis" primjenom steganografije.
%% Ulazni argument mov, tj. video zapis treba pripremiti za ovu funkciju nasljedeći način u npr.
%komandnom prozoru(eng.Command Window-u) od MATLAB-a :
% vidObj = VideoReader('Superman.mp4'); % ovo dolje smo napisali tako što smo gledali
% MATLAB-ovu dokumentaciju za funkciju hasFrame:
% vidHeight = vidObj.Height;
% vidWidth = vidObj.Width;
% mov = struct('cdata',zeros(vidHeight,vidWidth,3,'uint8'), 'colormap',[]);
% k = 1;
% while hasFrame(vidObj)
% mov(k).cdata = readFrame(vidObj);
% k = k+1;
% end
%% To gore treba otkomentarisati za video, ili napraviti novu funkciju.
% Slijedi nastavak za audio, a u tu svrhu treba u komandnom terminalu izvršiti sljedeće naredbe:
% [y,Fs] = audioread('mySpeech.wav'); % Učitamo snimljeni audio u MATLAB
% sound(y,Fs) % Poslušamo snimljeni audio da se uvjerimo da je to što treba biti, a potom taj y
% niz tipa double konvertujemo u binarni znakovni (eng.char) niz sa donjom naredbom:
% wavebinary = dec2bin( typecast( single(y), 'uint8'), 8 ) - '0';
% wavdata = reshape( typecast( uint8(bin2dec( char(wavebinary + '0') )), 'single' ), size(y) );
% Ovo gore je inverzna funkcija
% Nakon toga, pretvaramo ga u logički niz logicalWavebinary, tj. pravi binarni niz koji skrivamo.
% Pošto je double tip od wavebinary niza, trebamo porediti sa ==1 da pretvorimo u logički niz.
logicalWavebinary = (wavebinary == 1);% Nadalje, trebaju nam biti, tj. logički tip podataka
logWave = reshape(logicalWavebinary.',1,[]);% Vektor vrsta audio bita za serijski prenos i
%umetanje u video frejm
[m,n] = size(logicalWavebinary);
% Ako je audio .wav zapis imao trajanje od 10 sekundi sa Fs=44100, onda će wavebinary
% imati veličinu m=1761280 i n=8
[mm, nn] = size(mov);
% Kod nas za mov=Superman.mp4 je mm=1 i nn=436
[mmm, nnn, brBoja] = size(mov(1).cdata);
% U našem primjeruće biti mmm=480, nnn=640 i brBoja=3 jer je RGB frejm - pa imamo tri boje.
% vidHeight == mmm; vidWidth == 640 kod korišćenog video zapisa.
% OutputVideo je instanca video strukture u koju ćemo smjestiti skriveni .wav audio zapis
OutputVideo = struct('cdata',zeros(mmm,nnn,3,'uint8'), 'colormap',[]);
ukBrBita = m*n;% Ukupan broj bita u audio binarnom zapisu
ukupanBrVrstaFrejma = mmm*nnn;% U našem primjeru će biti 480*640=307200
potrebanBrStegoFrejmova = floor(ukBrBita/(brBoja*ukupanBrVrstaFrejma)); % Pomnožili
%smo sa brBoja=3 jer ovdje koristimo RGB video frejmove.
% Takođe, dijelimo sa 8 jer po jedan audio bit ide na svaki bajt od frejma, tj.
% LSBs (utiskujemo audio bite u bite najmanje težine video zapisa) steganography implementiramo
ostatakBita = ukBrBita - potrebanBrStegoFrejmova*brBoja*ukupanBrVrstaFrejma;
if ((ukupanBrVrstaFrejma*brBoja*mm*nn) < ukBrBita) % provjeravamo
disp('Video nije dovoljno dugacak za ubacivanje audio zapisa sa LSBs steganografijom');
disp('Treba Vam ulazni RGB video od minimalno ovoliko frejmova: ');
if (ostatakBita == 0)
disp(potrebanBrStegoFrejmova);
else
disp(potrebanBrStegoFrejmova + 1);
end
return;
end
for i=1:1:potrebanBrStegoFrejmova
I_frame_red = double(mov(i).cdata (:,:,1));
I_frame_green = double(mov(i).cdata (:,:,2));
I_frame_blue = double(mov(i).cdata (:,:,3));
% Sada, dolje pretvaramo frejmove iz tipadouble u binarne char zapise po 8 bita
ulazni_binarni_stream_red = dec2bin( I_frame_red, 8 );
ulazni_binarni_stream_green = dec2bin( I_frame_green, 8) ;
ulazni_binarni_stream_blue = dec2bin( I_frame_blue, 8) ;
% Gore smo pretvorili u pojedinačne binarne nizove koji su tipa char, a sadaih trebamo pretvoriti u
%tip logical, tj. zaista pravi binarni niz. U nastavku je to implementirano poređenjem sa znakom '1'
logicalFrejm_red = (ulazni_binarni_stream_red == '1');
logicalFrejm_green = (ulazni_binarni_stream_green == '1');
logicalFrejm_blue = (ulazni_binarni_stream_blue == '1');
for j=1:1:ukupanBrVrstaFrejma% Sa ove tri for petlje vršimo umetanje bita i
% prolazimo kroz sve vrste frejma, pa je zato doukupanBrVrstaFrejma
logicalFrejm_red(j,8) = logWave(1,(i-1)*brBoja*ukupanBrVrstaFrejma + j); % Ovdje samo u
%8. koloni frejma mijenjamo bite jer su to biti na poziciji najmanje težine – eng. LSBs
end
% Sada trebamo ovaj modifikovani frejm sačuvati kao stego frejm. Dakle, ponovo iz logičkog
```

```
% tipa se moramo vratiti u char tip i tu konverziju radimopomoću narednih komandi za RED –
% frejmove video zapisa, pa za GREEN frejmove itd:
StegoCharBinarniFrejm_red = char(logicalFrejm_red + '0');
InverzCharBinarniFrejm_red = uint8(bin2dec(StegoCharBinarniFrejm_red));
praviCharBinarniFrejm_red = reshape(InverzCharBinarniFrejm_red,size(I_frame_red));
% Potom, taj frejm memorišemo u izlazni video RED frejm na sljedeći način:
OutputVideo(i).cdata(:,:,1) = praviCharBinarniFrejm_red;
% Ponavljamo postupak za GREEN frejmove video zapisa u boji (sa RGB frejmovima)
for j=1:1:ukupanBrVrstaFrejma
logicalFrejm_green(j,8) = logWave(1,(i-1)*brBoja*ukupanBrVrstaFrejma + j + ukupanBrVrstaFrejma);
end
StegoCharBinarniFrejm_green = char(logicalFrejm_green + '0');
    InverzCharBinarniFrejm_green = uint8(bin2dec(StegoCharBinarniFrejm_green));
        praviCharBinarniFrejm_green= reshape(InverzCharBinarniFrejm_green,size(I_frame_green));
% Nadalje, taj frejm memorišemo u izlazni video frejm komponente GREEN na sljedeći način:
OutputVideo(i).cdata(:,:,2) = praviCharBinarniFrejm_green;
% Potom, ponavljamo postupak za BLUE frejmove od video zapisa
for j=1:1:ukupanBrVrstaFrejma
logicalFrejm_blue(j,8)=logWave(1,(i-1)*brBoja*ukupanBrVrstaFrejma + j + 2*ukupanBrVrstaFrejma);
end
StegoCharBinarniFrejm_blue = char(logicalFrejm_blue + '0');
InverzCharBinarniFrejm_blue = uint8(bin2dec(StegoCharBinarniFrejm_blue));
praviCharBinarniFrejm_blue= reshape(InverzCharBinarniFrejm_blue,size(I_frame_blue));
% Potom, memorišemo u izlazni video frejm komponente BLUE na sljedeći način:
OutputVideo(i).cdata(:,:,3) = praviCharBinarniFrejm_blue;
% Napokon smo gornjom komandom utisnuli audio (.wav) zapis u video frejmove svih komponenti
%RED, GREEN i BLUE kod RGB videa u boji.
end
if (ostatakBita == 0)
    % Moramo dodati i ostale frejmove iz mov u OutputVideo strukturu i to bitski nepromijenjeno!
for l=potrebanBrStegoFrejmova:1:nn% Prolazimo kroz ostale frejmove
        OutputVideo(l).cdata(:,:,:) = mov(l).cdata(:,:,:);
end
end
if ((ostatakBita >= 1) && (ostatakBita <= ukupanBrVrstaFrejma))
for j=1:1:ostatakBita
logicalFrejm_red(j,8) = logWave(1,potrebanBrStegoFrejmova*brBoja*ukupanBrVrstaFrejma + j);
end
StegoCharBinarniFrejm_red = char(logicalFrejm_red + '0');
InverzCharBinarniFrejm_red = uint8(bin2dec(StegoCharBinarniFrejm_red));
praviCharBinarniFrejm_red = reshape(InverzCharBinarniFrejm_red,size(I_frame_red));
OutputVideo(potrebanBrStegoFrejmova + 1).cdata(:,:,1) = praviCharBinarniFrejm_red;
% Potom, dodajemo ostale frejmove iz mov u OutputVideo kako bi mogli iskoristiti funkciju
% snimi_u_avi_format_video.m
 OutputVideo(potrebanBrStegoFrejmova + 1).cdata(:,:,2) = mov(potrebanBrStegoFrejmova + 1).cdata(:,:,2);
 OutputVideo(potrebanBrStegoFrejmova + 1).cdata(:,:,3) = mov(potrebanBrStegoFrejmova + 1).cdata(:,:,3);
for l=(potrebanBrStegoFrejmova + 2):1:nn% Prolazimo kroz ostale frejmove
OutputVideo(l).cdata(:,:,:) = mov(l).cdata(:,:,:);% Memorišemo
end
elseif ((ostatakBita > ukupanBrVrstaFrejma) && (ostatakBita <= 2*ukupanBrVrstaFrejma))
for j=1:1:ukupanBrVrstaFrejma
logicalFrejm_red(j,8) = logWave(1,potrebanBrStegoFrejmova*brBoja*ukupanBrVrstaFrejma + j);
    ostatakBita = ostatakBita - 1;
end
    % Sada trebamo ovaj modifikovani RED frejm sačuvati kao stego frejm
StegoCharBinarniFrejm_red = char(logicalFrejm_red + '0');
InverzCharBinarniFrejm_red = uint8(bin2dec(StegoCharBinarniFrejm_red));
praviCharBinarniFrejm_red = reshape(InverzCharBinarniFrejm_red,size(I_frame_red));
% Potom, taj frejm memorišemo u izlazni video frejm na sljedeći način
OutputVideo(potrebanBrStegoFrejmova + 1).cdata(:,:,1) = praviCharBinarniFrejm_red;
% U nastavku, dodamo neprimijenjene ostale bite kod GREEN frejmova video zapisa
for j=1:1:ostatakBita
        logicalFrejm_green(j,8)   =   logWave(1,potrebanBrStegoFrejmova*brBoja*ukupanBrVrstaFrejma + j + ukupanBrVrstaFrejma);
end
% Nadalje, ponavljamo postupak, iz logičkog moramo se vratiti u char tip za GREEN frejmove itd.
    StegoCharBinarniFrejm_green = char(logicalFrejm_green + '0');
InverzCharBinarniFrejm_green = uint8(bin2dec(StegoCharBinarniFrejm_green));
praviCharBinarniFrejm_green=          reshape(InverzCharBinarniFrejm_green,size(I_frame_green));
% Sada te frejmove memorišemo u izlazni video frejm na analogan način kao i prije
 OutputVideo(potrebanBrStegoFrejmova + 1).cdata(:,:,2) = praviCharBinarniFrejm_green;
OutputVideo(potrebanBrStegoFrejmova + 1).cdata(:,:,3) = mov(potrebanBrStegoFrejmova + 1).cdata(:,:,3);
for l=(potrebanBrStegoFrejmova + 2):1:nn% Prolazimo kroz ostale frejmove
OutputVideo(l).cdata(:,:,:) = mov(l).cdata(:,:,:);% Memorišemo nepromijenjene bite
end
else% Potom, dolje dodajemo nepromijenjene ostale bite kod RED frejmova
for j=1:1:ukupanBrVrstaFrejma
logicalFrejm_red(j,8) = logWave(1,potrebanBrStegoFrejmova*brBoja*ukupanBrVrstaFrejma + j);
ostatakBita = ostatakBita - 1;
end
% Nadalje, ponovo iz logical tipa moramokonvertovati u char tip kod RED frejmova sa:
StegoCharBinarniFrejm_red = char(logicalFrejm_red + '0');
InverzCharBinarniFrejm_red = uint8(bin2dec(StegoCharBinarniFrejm_red));
praviCharBinarniFrejm_red = reshape(InverzCharBinarniFrejm_red,size(I_frame_red));
% Potom,  taj RED frejm memorišemo u izlazni video frejm na sljedeći način:
OutputVideo(potrebanBrStegoFrejmova + 1).cdata(:,:,1) = praviCharBinarniFrejm_red;
for j=1:1:ukupanBrVrstaFrejma
logicalFrejm_green(j,8)=  logWave(1,potrebanBrStegoFrejmova*brBoja*ukupanBrVrstaFrejma + j + ukupanBrVrstaFrejma);
ostatakBita = ostatakBita - 1;
end
StegoCharBinarniFrejm_green = char(logicalFrejm_green + '0');
InverzCharBinarniFrejm_green = uint8(bin2dec(StegoCharBinarniFrejm_green));
praviCharBinarniFrejm_green=          reshape(InverzCharBinarniFrejm_green,size(I_frame_green));
OutputVideo(potrebanBrStegoFrejmova + 1).cdata(:,:,2) = praviCharBinarniFrejm_green;
for j=1:1:ostatakBita
  logicalFrejm_blue(j,8) = logWave(1,potrebanBrStegoFrejmova*brBoja*ukupanBrVrstaFrejma + j + 2*ukupanBrVrstaFrejma);
end
StegoCharBinarniFrejm_blue = char(logicalFrejm_blue + '0');
InverzCharBinarniFrejm_blue = uint8(bin2dec(StegoCharBinarniFrejm_blue));
praviCharBinarniFrejm_blue= reshape(InverzCharBinarniFrejm_blue,size(I_frame_blue));
OutputVideo(potrebanBrStegoFrejmova + 1).cdata(:,:,3) = praviCharBinarniFrejm_blue;
for l=(potrebanBrStegoFrejmova + 2):1:nn% Prolazimo kroz ostale frejmove
OutputVideo(l).cdata(:,:,:) = mov(l).cdata(:,:,:);% Memorišemo u izlazni video zapis
end
% Napokon smo gore utisnuli cijeli audio (.wav) zapis u video frejmove komponenti RGB.
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Listing 3.2: Izvorni MATLAB kod funkcije *NasaSteganografija.m*

Nadalje, radi potpunog razumijevanja rada funkcija u listingu 3.1, potrebno je da prikažemo listing 3.3 i opis naredne funkcije koju smo implementirali, tj. funkcije *snimi_u_avi_format_video.m*. Naime, pomoću te funkcije kreiramo nekompresovani video zapis (konkretno u našem slučaju *stegoSuperman.avi*) u koji je LSB steganografijom utisnut audio zapis.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function snimi_u_avi_format_video(OutputVideo)
%% Autori: Boris Malčić, Zoran Đurić, Slavko Šajić, ETFBL, UNIBL, RS/BiH
% Sva prava zadržavaju autori i nije dozvoljeno korištenje funkcija bez referenciranja ovog članka
% OutputVideo je izlaz funkcije NasaSteganografija.m
% Nećemo ulaziti u efikasnost jer moramo iz formata .mp4 snimiti u .avi nekompresovani format
% Naš video Superman.mp4, u nastavku možete zamijeniti svojim video zapisom formata .mp4, ali
reader = VideoReader('Superman.mp4'); % samo pripazite na veličinu video zapisa.
writer = VideoWriter('stegoSuperman.avi','Uncompressed AVI');% Mana – nekompresovan, tj.
% mora se koristiti nekompresovani .avi format, stegoSuperman.avi je izlazni stego-video zapis.
writer.FrameRate = reader.FrameRate;% Izjednačavamo broj frejmova u sekundi (npr. 30 fps)
open(writer);% Otvaramo objekat za snimanje izlaznog video zapisa – tj. stegoSuperman.avi
for kk=1:1:length(OutputVideo)% Prolazimo kroz sve frejmove
writeVideo(writer,OutputVideo(kk));% Upisujemo kk-ti frejm u izlazni stegoSuperman.avi
end
```

```
close(writer);% Na kraju, poslije upisivanja svih frejmova, zatvaramo video objekat writer.
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Listing 3.3: Izvorni MATLAB kod funkcije *snimi_u_avi_format_video.m*

Nadalje, na prijemu je potrebno izvršiti inverznu funkciju kojom ćemo iz *stegoSuperman.avi* video zapisa ekstrakovati audio zapis, odnosno realizovati inverznu steganografiju. Taj potupak smo implementirali funkcijom *InverzSteganografija.m* čija je implementacija i opis data u listingu 3.4.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [wavdata, dec_DES_wavebinary] = InverzSteganografija(r_mov, y_format )
%% Autori: Boris Malčić, Zoran Đurić, Slavko Šajić
%    Elektrotehnički fakultet, Univerzitet u Banjoj Luci, Republika Srpska, BiH
%    Sada je r_mov ulazni argument (eng. received movie – r_mov)
%    Izlaz funkcije je wavdata koji se može poslušati sa:
% sound(wavdata,Fs); % gdje je Fs=44100 frekvencija odmjeravanja audio zapisa
% y_format je format audio zapisa, tj. dobija se kada se u komandnom prozoru MATLAB-a izvrše
% sljedeće naredbe:
% SnimiFormatSpeech % Pozivamo funkciju da snimimoaudio zapis koji služi kao format, a potom:
% [y_format,Fs] = audioread('formatSpeech.wav'); % Učitamo snimljeni audio u MATLAB
%% sound(y_format,Fs) % Poslušamo snimljeni audio zapis da se uvjerimo da je u redu.
format_wavebinary = dec2bin(typecast(single(y_format(:)), 'uint8'), 8 ) - '0';% Konvertujemo
% Takođe, kao ulaz u ovu funkciju se mora znati kada počinje skriveni audio i kada završava, ili
% prosto, neka je unaprijed dogovoreno da se skriva fiksna dužina .wav zapisa trajanja nekoliko
% sekundi i da počinje od početka r_mov skrivanje/utiskivanje sa LSB steganografskom metodom
% Prema tome, zato funkcija prima format_wavebinary kao ulazni argument, tj. mora unaprijed da
% zna format u kojem je pravi skriven pravi audio zapis koji će ekstrakovati.
% Za snimanje audio zapisa pomoću mikrofona smo koristili MATLAB primjer iz dokumentacije:
%Read from Microphone and Write to Audio File.m
% Izlaz te funkcije iz MATLAB-ovog primjera, koju smo modifikovali u svoje svrhe, bioje
% mySpeech.wavaudio zapis koji smo utisnuli u video zapis pomoću naše implementacije LSB
% steganografije. Nadalje, ulazni argument mov, tj. video zapis, treba pripremiti na sljedeći način u
% komandnom prozoru:
% vidObj = VideoReader('Superman.mp4'); % Pogledajte dokumentaciju za MATLAB – ovu
% funkciju pod nazivom hasFrame
nizBita_wavbinary = reshape(format_wavebinary.',1,[]);
stvarniBinarniNiz_wavbinary = dec2bin(nizBita_wavbinary,1);
logWave = (stvarniBinarniNiz_wavbinary == '1');% Trebaju nam biti, tj. logički tip podataka
d_logWave = logWave;% Ovdje će biti ekstrakovani logički biti od skrivenog audia iz videa
[m,n] = size(format_wavebinary);% Poznat nam je format audio zapisa koji trebamo ekstrakovati
[mm, nn] = size(r_mov);
[mmm, nnn, brBoja] = size(r_mov(1).cdata);
ukBrBita = m*n;% Ukupan broj bita u audio binarnom zapisu
ukupanBrVrstaFrejma = mmm*nnn;% Kod nas će u ovom primjeru biti 480*640=307200
potrebanBrStegoFrejmova = floor(ukBrBita/(brBoja*ukupanBrVrstaFrejma)); % Treba nam
% floor za zaokruživanje.Pomnožili smo sa brBoja=3 jer ovdje koristimo RGB video frejmove
% Dijelimo sa 8 - jer po jedan audio bit ide na svaki bajt od frejma jer implementiramo LSB
% steganografiju.
ostatakBita = ukBrBita - potrebanBrStegoFrejmova*brBoja*ukupanBrVrstaFrejma;
if ((ukupanBrVrstaFrejma*brBoja*mm*nn) < ukBrBita)
disp('Primljeni video nije dovoljno dugacak za ekstrakciju audio zapisa sa LSBs steganografijom');
    disp('Treba Vam ulazni RGB video od minimalno ovoliko frejmova: ');
if (ostatakBita == 0)
    disp(potrebanBrStegoFrejmova);
else
      disp(potrebanBrStegoFrejmova + 1);% Inače, potreban broj frejmova u videu
end
return;
end
for i=1:1:potrebanBrStegoFrejmova
    I_frame_red = double(r_mov(i).cdata (:,:,1));
    I_frame_green = double(r_mov(i).cdata (:,:,2));
    I_frame_blue = double(r_mov(i).cdata (:,:,3));
% Nadalje, pretvaramo double u binarne char zapise po 8 bita pomoću narednih komandi:
ulazni_binarni_stream_red = dec2bin( I_frame_red, 8 );
ulazni_binarni_stream_green = dec2bin( I_frame_green, 8) ;
ulazni_binarni_stream_blue = dec2bin( I_frame_blue, 8) ;
% Gore smo pretvorili u pojedinačne binarne nizove koji su tipa char, a u nastavku ih trebamo
% pretvoriti u tip logical, tj. zaista pravi binarni niz
logicalFrejm_red = (ulazni_binarni_stream_red == '1');
logicalFrejm_green = (ulazni_binarni_stream_green == '1');
logicalFrejm_blue = (ulazni_binarni_stream_blue == '1');
for j=1:1:ukupanBrVrstaFrejma% Sa naredne tri for petlje vršimo ekstrakciju (ek.) audio bita
d_logWave((i-1)*brBoja*ukupanBrVrstaFrejma + j,1) = logicalFrejm_red(j,8);% ek.
end
for j=1:1:ukupanBrVrstaFrejma
    d_logWave((i-1)*brBoja*ukupanBrVrstaFrejma + j + ukupanBrVrstaFrejma,1) = logicalFrejm_green(j,8);
end
for j=1:1:ukupanBrVrstaFrejma
d_logWave((i-1)*brBoja*ukupanBrVrstaFrejma + j + 2*ukupanBrVrstaFrejma,1) = logicalFrejm_blue(j,8);
end
end
if ((ostatakBita >= 1) && (ostatakBita <= ukupanBrVrstaFrejma))
for j=1:1:ostatakBita
     d_logWave(potrebanBrStegoFrejmova*brBoja*ukupanBrVrstaFrejma + j,1) = logicalFrejm_red(j,8);
end
elseif ((ostatakBita > ukupanBrVrstaFrejma) && (ostatakBita <= 2*ukupanBrVrstaFrejma))
for j=1:1:ukupanBrVrstaFrejma
     d_logWave(potrebanBrStegoFrejmova*brBoja*ukupanBrVrstaFrejma + j,1) = logicalFrejm_red(j,8);
   ostatakBita = ostatakBita - 1;
end
for j=1:1:ostatakBita
    d_logWave(potrebanBrStegoFrejmova*brBoja*ukupanBrVrstaFrejma + j + ukupanBrVrstaFrejma, 1) = logicalFrejm_green(j, 8);
end
else
for j=1:1:ukupanBrVrstaFrejma
     d_logWave(potrebanBrStegoFrejmova*brBoja*ukupanBrVrstaFrejma + j, 1) = logicalFrejm_red(j, 8);
   ostatakBita = ostatakBita - 1;
end
for j=1:1:ukupanBrVrstaFrejma
    d_logWave(potrebanBrStegoFrejmova*brBoja*ukupanBrVrstaFrejma + j + ukupanBrVrstaFrejma, 1) = logicalFrejm_green(j, 8);
   ostatakBita = ostatakBita - 1;
end
for j=1:1:ostatakBita
    d_logWave(potrebanBrStegoFrejmova*brBoja*ukupanBrVrstaFrejma + j + 2*ukupanBrVrstaFrejma, 1) = logicalFrejm_blue(j, 8);
end
end
testDouble_wavbinary = double(d_logWave + 0);
primljeniDouble_wavbinary = reshape(testDouble_wavbinary, 8, []);
stvarniPrimljeniDouble_wavbinary = primljeniDouble_wavbinary';
wavdata = reshape( typecast( uint8(bin2dec( char(stvarniPrimljeniDouble_wavbinary + '0') )), 'single' ), size(y_format) );
dec_DES_wavebinary = dec2bin( typecast( single(wavdata), 'uint8'), 8 ) - '0';
% Kada budemo enkriptovali sa DES algoritmom zatrebaće nam izlaz dec_DES_wavebinary.
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Listing 3.4: Izvorni MATLAB kod funkcije *InverzSteganografija.m*

Nadalje, potrebno je da prikažemo simulacione MATLAB – ove funkcije u izvornom obliku za drugo zeleno dugme prikazano na slici 3. Taj izvorni kod je predstavljen listingom 3.5.

```
%%%%%    Izvorni kod funkcije za 2. zeleno dugme sa slike 3. je sljedeći%%%%%%
function steganography_using_AES_256_bit_key(app, event)
%% Autori: M.Sc. Boris Malcic, Dr. Zoran Djuric, Dr. Slavko Sajic – NAPOMENA!
% NAPOMENA:Veći dio ove AES implementacije je preuzet od David Hill-a i dostupan online na:
%https://www.mathworks.com/matlabcentral/fileexchange/73412-advanced-encryption-standard-aes-128-192-256
clear all;close all;clc;% Objašnjeno prije.
tic; % Startujemo vremenski brojač kako bi izmjerili vrijeme izvršavanja do naredbe toc
[y,Fs] = audioread('mySpeech.wav');
sound(y,Fs)
```

```
wavebinary = dec2bin( typecast( single(y), 'uint8'), 8 ) - '0';
vidObj = VideoReader('Superman.mp4');
vidHeight = vidObj.Height;
vidWidth = vidObj.Width;
mov = struct('cdata',zeros(vidHeight,vidWidth,3,'uint8'), 'colormap',[]);
k = 1;
while hasFrame(vidObj)
mov(k).cdata = readFrame(vidObj);
k = k+1;
end
% U nastavku smo koristili i modifikovali AES funkcije od David-a Hill-a sa MathWorks-a
key='000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f';
In_hex = conversionTypeForAES(wavebinary); % Naša implementacija funkcije za konverziju
OutCipherAES = cipherForAES(In_hex,key); % Vršimo AES šifrovanje (enkripciju)
OutWavebinary = conversionTypeForStego(OutCipherAES); % Vršimo rekonverziju tipa
% Gore imamo AES enkriptovan OutWavebinary i sada sa tim pozivamo steganografiju
OutputVideo = NasaSteganografija(OutWavebinary, mov);
snimi_u_avi_format_video(OutputVideo);
% Potom, trebamo na prijemu iz steganografski izmjenjenog video zapisa ekstraktovati audio
RXvidObj = VideoReader('stegoSuperman.avi');
vidHeight = RXvidObj.Height;
vidWidth = RXvidObj.Width;
r_mov = struct('cdata',zeros(vidHeight,vidWidth,3,'uint8'), 'colormap',[]);
k = 1;
while hasFrame(RXvidObj)
        r_mov(k).cdata = readFrame(RXvidObj);
k = k+1;
end
[y_format,Fs] = audioread('myFormatSpeech.wav');% Učitamo snimljeni audio format
sound(y_format,Fs) % Poslušamo audio format da se uvjerimo da to nije enkriptovani audio
[wavdata, encAESwavebinary] = InverzSteganografija(r_mov, y_format);
sound(wavdata, Fs);% Poslušamo ovaj signal koji se treba čuti kaošum jer je AES enkriptovan
% Potom,u nastavku vršimo konverziju tipa i poslije toga dekripciju AES algoritma
RX_In_hex = conversionTypeForAES(encAESwavebinary); % Naša funkcija za konverziju
% tipova, tj. za prilagođenje kako bi mogli koristiti AES implementaciju sa našom implementacijom
invOutCipherAES = invCipherForAES(RX_In_hex,key); % Sada dešifrujemo AES
rxOutWavebinary = conversionTypeForStego(invOutCipherAES); % Ponovo konverzija
encAES_wavdata = reshape( typecast( uint8(bin2dec( char(rxOutWavebinary + '0') )), 'single'
), size(y_format) ); % Da dobijemo u potrebnom formatu .wav audio zapisa
sound(encAES_wavdata,Fs); % Konačno, poslušamo dekriptovani audio zapis.
toc; % Zaustavljamo vremenski brojač i vrijeme izvršavanja ispisujemo u komandnom prozoru.
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Listing 3.5: Izvorni kd funkcije *steganography_using_AES_256_bit_key.m* [8]

Listing 3.5 simulira AES enkriptovanje audio zapisa koji se uz to još utiskuje LSB steganografijom u video zapis. U cilju potpunog razumijevanja te simulacije, prikazaćemo listing funkcija koje se tu koriste. Prva od tih funkcija je *conversionTypeForAES.m*, čiji je implementacija i opis data u listingu 3.6. Nakon toga, poziva se funkcija *cipherForAES.m* data listingom 3.7, a kojom vršimo blokovsku AES enkripciju nad 128-bitnim ulaznim heksadecimalnim blokovima (označen sa *eng.* ***In_hex*** u listingu 3.5) audio zapisa i korišćenjem 256-bitnog ključa (označenog sa *eng.* ***key*** u listingu 3.5).

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function In_hex = conversionTypeForAES(wavebinary)
%% Autori: Boris Malčić, Zoran Đurić, Slavko Šajić
% ETFBL, UNIBL, RS/BiH
% In_hex - je izlaz ove funkcije koji treba biti ulazni podatak za AES algoritam, tj.  podatak u
% heksadecinalnom string zapisu
% wavebinary – je ulaz, tj. double matrica od audio zapisa koji smo koristili u steganografiji
% Poslije ćemo morati In_hex pretvoriti ponovo u tip wavebinary jer takav tip matrice prima naš
% algoritam za steganografiju
%%
logicalWavebinary = (wavebinary == 1);% Trebaju nam biti, tj. logički tip podataka
logWave = reshape(logicalWavebinary.',1,[]); % Vektor vrsta audio bita za serijski prenos i
% umetanje u frejm
In_hex = '1';
for i=1:4:(length(logWave)-3)
   testBr = logWave(1,i:(i+3));
   charTestBr = char(testBr + '0');
   decTest = bin2dec(charTestBr);
   hexTest = dec2hex(decTest);
   In_hex = [In_hex hexTest];
end
In_hex = In_hex(2:length(In_hex));
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Listing 3.6: Implementacija funkcije *conversionTypeForAES.m*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function OutCipherAES = cipherForAES(In_hex,key)
OutCipherAES = ''; % Izlaz je AES šifrat nad blokovima podataka po 32 heksadecimalne cifre
for i = 1:32:(length(In_hex)-31)% Dijelimo ulazni audio zapis na 32 hex. blokove
In = In_hex(i:(i+31));% Svaki blok ulazne plaintext poruke je 32*4 = 128 bita kod AES,
% tj. veličina bloka ulaznih podataka jer 32 hex cifre i svaka heksadecimalna cifra je 4 bita
Out = Cipher(key,In); % AES blokovski šifrat
OutCipherAES = [OutCipherAES Out]; % Konkatenacija pojedinačnih šifrata čini AES cjelinu
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Listing 3.7: Implementacija AES blokovskog šifrata – odnosno funkcije *cipherForAES.m*

Nadalje, razmatramo i dalje listing 3.5, gdje je poslije poziva i izvršavanja komande za AES blokovsko šifrovanje (tj. *cipherForAES.m*), ponovo potrebno izvršiti konverziju tipa u oblik pogodan za poziv steganografije – što je implementirano funkcijom *conversionTypeForStego.m* čiji izvorni kd i opis dajemo sa listingom 3.8.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function OutWavebinary = conversionTypeForStego(OutChiperAES)
% Ulazni argument OutChiperAES je izlaz funkcije chiperForAES.m
% Izlaz ove funkcije - OutWavebinary je ulazni argument za funkciju NasaSteganografija.m
LogOutWavebinary = (1 == 1);% Prvi (1.) logički broj je izabran proizvoljno (nebitan je)
for i=1:length(OutChiperAES)
binBr = dec2bin(hex2dec(OutChiperAES(i)),4);% Binarna reprezentacija sa najmanje 4 bita
logBr = (binBr == '1111');% Logički zapis dobijamo komparacijom sa '1111'
LogOutWavebinary = [LogOutWavebinary logBr];
End% U nastavku slijede potrebne komande za konverziju iz logičkog u binarni tip audio zapisa
LogOutWavebinary = LogOutWavebinary(2:(4*length(OutChiperAES) + 1));% Odbaciti 1.
testDouble_wavbinary = double(LogOutWavebinary + 0);
primljeniDouble_wavbinary = reshape(testDouble_wavbinary,8,[]);
OutWavebinary = primljeniDouble_wavbinary';% Dobijamo binarni zapis za steganografiju
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Listing 3.8: Implementacija funkcije *conversionTypeForStego.m*

Nadalje, izlaz *OutWaveBinary* iz funkcije *conversionTypeForStego.m* se koristi u listing 3.5 kao jedan od ulaznih argumenata pri pozivu funkcije *NasaSteganografija.m*, a drugi ulazni argument pri pozivu ove funkcije je označen sa *mov* i predstavlja identifikator video zapisa (kod nas *Superman.mp4*) čiji se biti modifikuju primjenom naše implementacije LSB steganografije u skladu sa bitima audio zapisa, tj. u skladu sa *OutWaveBinary*. Radi potpunog razumijevanja listinga 3.5 ostaje nam još da objasnimo i prikažemo izvorni kod funkcije *invCipherForAES.m* čiji je listing 3.9.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function invOutChiperAES = invChiperForAES(RX_In_hex, key)
invOutChiperAES = ''; % Prvi dekriptovan znak je prazan string. Dolje vršimo AES dekripciju
for i = 1:32:(length(RX_In_hex) - 31) % Prolazimo kroz sve 128-bitne AES šifrovane blokove
In1 = RX_In_hex(1,i:1:(i+31)); % Uzimamo i-ti ulazni AES šifrat (koji je dužine 32 hex znaka)
Out1 = InvCipher(key,In1); % Vršimo AES dešifrovanje/dekripciju nad i-tim blokom
invOutChiperAES = [invOutChiperAES Out1];%Vršimo konkatenaciju dekriptovanih blokova
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Listing 3.9: Implementacija funkcije *invCipherForAES.m*

U nastavku, potrebno je objasniti treću kriptografsku opciju, odnosno treće zeleno dugme sa slike 3 pod nazivom *Steganography with DES using 56-bit key for encryption/decryption of source audio record hidden in video movie*. Dakle, izvorni kd funkcija koje aktivira klik na 3. dugme, prikazan je sa listingom 3.10. Za razumijevanje rada ove simulacije, najvažnije je objasniti redom funkcije: 1. *conversionTypeForDES.m* kojom se vrši konverzija audio zapisa iz binarnog oblika u heksadecimalni string koji je potreban za poziv funkcije za blokovsku DES enkripciju; 2. *myDES_Enc.m* pomoću koje se vrši blokovska enkripcija sa DES algoritmom; 3. funkcija *myDES_Dec.m* kojom se vrši blokovska DES dekripcija na prijemu i 4. funkcija *conversionTypeForStego.m* kojom se vrši konverzija audio zapisa u tip pogodan da se izvrše ovdje implementirane funkcija za steganografiju i inverzne operacije na prijemu. Za prethodne četiri funkcije, izvorne kdove i njihov opis implementacije prikazujemo redom redom u listinzima 3.11, 3.12, 3.13 i 3.14 – respektivno. Takođe, vrlo je jednostavno korišćenjem prethodnih funkcija implementirati TDEA, a ali to ovdje nije implementirano zbog trostukog usporavanja vremena izvršavanja primjenom TDEA.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Izvorni kod funkcija koje se pozivaju klikom na 3. zeleno dugme GUI app sa slike 3%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function encrypDecrypt_audio_over_video_using_DES_and_LSB_steganography(app, event)
clear all;close all;clc;tic;% Već objašnjeno u našim prethodnim izvornim kodovima.
[y,Fs] = audioread('mySpeech.wav');% Snimimo neku kraću audio poruku, npr. riječ „TAJNA"
sound(y,Fs)% Poslušamo snimljeni audio. Poželjan je kraći audio zapis radi bržeg izvršavanja!
wavebinary = dec2bin( typecast( single(y), 'uint8'), 8 ) –'0';% Već prije objašnjeno
rx_In_hex = conversionTypeForDES(wavebinary);% Konverzija tipa radi primjene DES
[cipher_text, all_KEYs] = myDES_Enc(rx_In_hex);% Vršimo DES enkripciju
[plain_text] = myDES_Dec(cipher_text);% Vršimo DES dekripciju radi provjere u nastavku ...
% Sada ćemo provjeriti da li je plain_text == rx_In_hex
% Ako jeste, onda smo tačno odradili DES enkripciju i dekripciju
sum(plain_text == rx_In_hex);
size(plain_text);
size(rx_In_hex);% Vidimo da je sve tačno! Prema tome, možemo krenuti u steganografiju
OutWavebinary = conversionTypeForStego(cipher_text); % Konverzija tipova, kao i prije
vidObj = VideoReader('Superman.mp4');% Ovi postupci/komande su prije objašnjavani
vidHeight = vidObj.Height;
vidWidth = vidObj.Width;
mov = struct('cdata',zeros(vidHeight,vidWidth,3,'uint8'), 'colormap',[]);
k = 1;
while hasFrame(vidObj)
mov(k).cdata = readFrame(vidObj);
k = k+1;
end
OutputVideo = NasaSteganografija(OutWavebinary, mov); % Vršimo steganografiju
snimi_u_avi_format_video(OutputVideo);
% Sada trebamo na prijemu iz stego video zapisa„izvući – tj. ekstrakovati" audio zapis
RXvidObj = VideoReader('stegoSuperman.avi');
vidHeight = RXvidObj.Height;
vidWidth = RXvidObj.Width;
r_mov = struct('cdata',zeros(vidHeight,vidWidth,3,'uint8'), 'colormap',[]);
k = 1;
while hasFrame(RXvidObj)
r_mov(k).cdata = readFrame(RXvidObj);
k = k+1;
end
[y_format,Fs] = audioread('myFormatSpeech.wav'); % Učitamo format audio zapisa
sound(y_format,Fs)% Poslušamo
[wavdata, dec_DES_wavebinary] = InverzSteganografija(r_mov, y_format);
sound(wavdata,Fs);% Čujemošum jer je to DES enkriptovan audio zapis
received_rx_In_hex = conversionTypeForDES(dec_DES_wavebinary);
        % Ako je sve u redu, onda sada received_rx_In_hex treba biti jednako cipher_text
        % Da provjerimo:
[primljeni_plain_text] = myDES_Dec(received_rx_In_hex);% DES dekripcija na prijemu
primljeni_OutWavebinary = conversionTypeForStego(primljeni_plain_text);
primljeni_wavdata = reshape(typecast(uint8(bin2dec(char(primljeni_OutWavebinary +
'0') )), 'single' ), size(y_format) );
sound(primljeni_wavdata, Fs);% Ako je sve u redu, čućemo snimljenu audio poruku "TAJNA".
toc;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Listing 3.10: Implementacija i opis funkcije *encrypDecrypt_audio_over_video_using_DES_and_LSB_steganography.m*

```
%% Mnoge implementacije smo realizovali analogno kao kod prethodnog opisa AES-a
function rx_In_hex = conversionTypeForDES(dec_DES_wavebinary)
logicalWavebinary = (dec_DES_wavebinary == 1);% Trebaju nam biti, tj. logički tip podataka
logWave = reshape(logicalWavebinary.',1,[]);% Vektor audio bita za serijski prenos i utiskivanje
rx_In_hex = '1';
for i=1:4:(length(logWave)-3)
   testBr = logWave(1,i:(i+3));
   charTestBr = char(testBr + '0');
   decTest = bin2dec(charTestBr);
   hexTest = dec2hex(decTest);
   rx_In_hex = [rx_In_hex hexTest];
end
rx_In_hex = rx_In_hex(2:length(rx_In_hex));
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Listing 3.11: Implementacija funkcije *conversionTypeForDES.m*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Donja DES implementacija je napisana korišćenjem više online dostupnih DES implementacija,
% kao što je npr. online implementacija od: http://freesourcecode.net/matlabprojects/59871/data-encryption-standard-%28des%29-in-matlab#.Y4JP93bMI2w
function [cipher_text, all_KEYs] = myDES_Enc(rx_In_hex)
cipher_text = [];% Šifrat
all_KEYs = [];% Koristićemo različite ključeve za šifrovanje pojedinih blokova
% Ulazni argument se dobija donjim komandama u komandnom prozoru MATLAB-a
% [y,Fs] = audioread('mySpeech.wav');
% sound(y,Fs)
% wavebinary = dec2bin( typecast( single(y), 'uint8'), 8 ) - '0';
% rx_In_hex = conversionTypeForDES(wavebinary);
% Ove donje funkcije za implementaciju DES-a nismo mi realizovali, već su preuzete sa Internet-a
for k=1:16:(length(rx_In_hex) - 15)
   M = rx_In_hex(k:(k + 15));
MB=[];
for i=1:16
Mi=M(i);
MBi=['0000',dec2bin(hex2dec(Mi))];
MBi=MBi(end-3:end);
MBi=[str2num(MBi(1)),str2num(MBi(2)),str2num(MBi(3)),str2num(MBi(4))];
MB=[MB,MBi];
end
M=MB;
K = '1256984563214569';
KB=[];
for i=1:16
   Ki=K(i);
   KBi=['0000',dec2bin(hex2dec(Ki))];
   KBi=KBi(end-3:end);
   KBi=[str2num(KBi(1)),str2num(KBi(2)),str2num(KBi(3)),str2num(KBi(4))];
   KB=[KB,KBi];
end
```

```
K=KB;
all_KEYs = [all_KEYs K];
E=[32, 1, 2, 3, 4, 5; 4, 5, 6, 7, 8, 9; 8, 9,10,11,12,13; 12,13,14,15,16,17; 16,17,18,19,20,21;
20,21,22,23,24,25; 24,25,26,27,28,29; 28,29,30,31,32,1];
S1=[14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5, 9, 0, 7; 0, 15, 7, 4, 14, 2, 13, 1, 10, 6, 12, 11, 9, 5, 3, 8;
4, 1, 14, 8, 13, 6, 2, 11, 15, 12, 9, 7, 3, 10, 5, 0; 15, 12, 8, 2, 4, 9, 1, 7, 5, 11, 3, 14, 10, 0, 6, 13];
S2=[15, 1, 8, 14, 6, 11, 3, 4, 9, 7, 2, 13, 12, 0, 5, 10; 3, 13, 4, 7, 15, 2, 8, 14, 12, 0, 1, 10, 6, 9, 11, 5;
0, 14, 7, 11, 10, 4, 13, 1, 5, 8, 12, 6, 9, 3, 2, 15; 13, 8, 10, 1, 3, 15, 4, 2, 11, 6, 7, 12, 0, 5, 14, 9];
S3=[10, 0, 9, 14, 6, 3, 15, 5, 1, 13, 12, 7, 11, 4, 2, 8; 13, 7, 0, 9, 3, 4, 6, 10, 2, 8, 5, 14, 12, 11, 15, 1;
 13, 6, 4, 9, 8, 15, 3, 0, 11, 1, 2, 12, 5, 10, 14, 7; 1, 10, 13, 0, 6, 9, 8, 7, 4, 15, 14, 3, 11, 5, 2, 12];
S4=[7, 13, 14, 3, 0, 6, 9, 10, 1, 2, 8, 5, 11, 12, 4, 15; 13, 8, 11, 5, 6, 15, 0, 3, 4, 7, 2, 12, 1, 10, 14, 9;
 10, 6, 9, 0, 12, 11, 7, 13, 15, 1, 3, 14, 5, 2, 8, 4; 3, 15, 0, 6, 10, 1, 13, 8, 9, 4, 5, 11, 12, 7, 2, 14];
S5=[2, 12, 4, 1, 7, 10, 11, 6, 8, 5, 3, 15, 13, 0, 14, 9; 14, 11, 2, 12, 4, 7, 13, 1, 5, 0, 15, 10, 3, 9, 8, 6;
 4, 2, 1, 11, 10, 13, 7, 8, 15, 9, 12, 5, 6, 3, 0, 14; 11, 8, 12, 7, 1, 14, 2, 13, 6, 15, 0, 9, 10, 4, 5, 3];
S6=[12, 1, 10, 15, 9, 2, 6, 8, 0, 13, 3, 4, 14, 7, 5, 11; 10, 15, 4, 2, 7, 12, 9, 5, 6, 1, 13, 14, 0, 11, 3, 8;
 9, 14, 15, 5, 2, 8, 12, 3, 7, 0, 4, 10, 1, 13, 11, 6; 4, 3, 2, 12, 9, 5, 15, 10, 11, 14, 1, 7, 6, 0, 8, 13];
S7=[4, 11, 2, 14, 15, 0, 8, 13, 3, 12, 9, 7, 5, 10, 6, 1; 13, 0, 11, 7, 4, 9, 1, 10, 14, 3, 5, 12, 2, 15, 8, 6;
 1, 4, 11, 13, 12, 3, 7, 14, 10, 15, 6, 8, 0, 5, 9, 2; 6, 11, 13, 8, 1, 4, 10, 7, 9, 5, 0, 15, 14, 2, 3, 12];
S8=[13, 2, 8, 4, 6, 15, 11, 1, 10, 9, 3, 14, 5, 0, 12, 7; 1, 15, 13, 8, 10, 3, 7, 4, 12, 5, 6, 11, 0, 14, 9, 2;
 7, 11, 4, 1, 9, 12, 14, 2, 0, 6, 10, 13, 15, 3, 5, 8; 2, 1, 14, 7, 4, 10, 8, 13, 15, 12, 9, 0, 3, 5, 6, 11];
P=[16,7,20,21,29,12,28,17,1,15,23,26,5,18,31,10,2,8,24,14,32,27,3,9,19,13,30,6,22,11,4,25];
PC1=[57,49,41,33,25,17,9,1,58,50,42,34,26,18,10,2,59,51,43,35,27,19,11,3,60,52,44,36,63,55,47,3
9,31,23,15,7,62,54,46,38,30,22,14,6,61,53,45,37,29,21,13,5,28,20,12,4];
PC2=[14,17,11,24,1,5,3,28,15,6,21,10,23,19,12,4,26,8,16,7,27,20,13,2,41,52,31,37,47,55,30,40,51,
45,33,48,44,49,39,56,34,53,46,42,50,36,29,32];
Ki=zeros(16,48);
K_PC1=K(PC1);
C0=K_PC1(1:28);
D0=K_PC1(29:56);
for i=1:16
if i==1||i==2||i==9||i==14||i==16
       C0=[C0(2:end),C0(1)];
       D0=[D0(2:end),D0(1)];
else
       C0=[C0(3:end),C0(1:2)];
       D0=[D0(3:end),D0(1:2)];
end
   K_LS=[C0,D0];
   Ki(i,:)=K_LS(PC2);
end
L=M(1:32);
R=M(33:64);
for i=1:16
   E0=reshape(E',1,48);
   R_E=R(E0);
   R_Ki=mod(R_E+Ki(i,:),2);
   B=R_Ki(1:6);
   x=B(1)*2+B(6)+1;
   y=B(2)*8+B(3)*4+B(4)*2+B(5)+1;
   C=['0000',dec2bin(S1(x,y))];
   C=C(end-3:end);
   C1=[str2num(C(1)),str2num(C(2)),str2num(C(3)),str2num(C(4))];
   B=R_Ki(7:12);
   x=B(1)*2+B(6)+1;
   y=B(2)*8+B(3)*4+B(4)*2+B(5)+1;
   C=['0000',dec2bin(S2(x,y))];
  C=C(end-3:end);
   C2=[str2num(C(1)),str2num(C(2)),str2num(C(3)),str2num(C(4))];
      B=R_Ki(13:18);
x=B(1)*2+B(6)+1;
   y=B(2)*8+B(3)*4+B(4)*2+B(5)+1;
   C=['0000',dec2bin(S3(x,y))];
   C=C(end-3:end);
   C3=[str2num(C(1)),str2num(C(2)),str2num(C(3)),str2num(C(4))];
   B=R_Ki(19:24);
   x=B(1)*2+B(6)+1;
   y=B(2)*8+B(3)*4+B(4)*2+B(5)+1;
   C=['0000',dec2bin(S4(x,y))];
   C=C(end-3:end);
   C4=[str2num(C(1)),str2num(C(2)),str2num(C(3)),str2num(C(4))];
  B=R_Ki(25:30);
   x=B(1)*2+B(6)+1;
   y=B(2)*8+B(3)*4+B(4)*2+B(5)+1;
   C=['0000',dec2bin(S5(x,y))];
   C=C(end-3:end);
   C5=[str2num(C(1)),str2num(C(2)),str2num(C(3)),str2num(C(4))];
      B=R_Ki(31:36);
   x=B(1)*2+B(6)+1;
   y=B(2)*8+B(3)*4+B(4)*2+B(5)+1;
   C=['0000',dec2bin(S6(x,y))];
   C=C(end-3:end);
   C6=[str2num(C(1)),str2num(C(2)),str2num(C(3)),str2num(C(4))];
   B=R_Ki(37:42);
x =B(1)*2+B(6)+1;
   y=B(2)*8+B(3)*4+B(4)*2+B(5)+1;
   C=['0000',dec2bin(S7(x,y))];
   C=C(end-3:end);
   C7=[str2num(C(1)),str2num(C(2)),str2num(C(3)),str2num(C(4))];
   B=R_Ki(43:48);
x =B(1)*2+B(6)+1;
   y=B(2)*8+B(3)*4+B(4)*2+B(5)+1;
   C=['0000',dec2bin(S8(x,y))];
   C=C(end-3:end);
   C8=[str2num(C(1)),str2num(C(2)),str2num(C(3)),str2num(C(4))];
   C=[C1,C2,C3,C4,C5,C6,C7,C8];
   R_P=C(P);
   TEMP=L;
   L=R;
   R=mod(TEMP+R_P,2);
end
TEMP=L;
L=R;
R=TEMP;
C=[L,R];
CS=[];
C=num2str(C);
pos=find(C~=' ');
C=C(pos);
for i=1:4:61
  Ci=C(i:i+3);
  CS=[CS,num2str(dec2hex(bin2dec(Ci)))];
end
C=CS;
cipher_text = [cipher_text C];
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Listing 3.12: Implementacija ***myDES_Enc.m*** je modifikacija online dostupnih funkcija [9-10]

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [plain_text] = myDES_Dec(cipher_text)
plain_text = [];
for k=1:16:(length(cipher_text) - 15)
M = cipher_text(k:(k + 15));
MB=[];
for i=1:16
Mi=M(i); % Ulazne poruke se dijele na blokove po 16 hex znakova, tj. 16*4=64 bita
MBi=['0000', dec2bin(hex2dec(Mi))];
MBi=MBi(end-3:end);
MBi=[str2num(MBi(1)), str2num(MBi(2)), str2num(MBi(3)), str2num(MBi(4))];
MB=[MB, MBi];
end
M=MB ;
K = '1256984563214569'; % Ključ je dužine 16 hex. znakova, odnosno 16*4 = 64 bita
KB=[];
for i=1:16
Ki=K(i);
KBi=['0000',dec2bin(hex2dec(Ki))];
 KBi=KBi(end-3:end);
KBi=[str2num(KBi(1)), str2num(KBi(2)), str2num(KBi(3)), str2num(KBi(4))];
KB=[KB, KBi];
end
K=KB;
E=[32, 1, 2, 3, 4, 5;4, 5, 6, 7, 8, 9; 8, 9,10,11,12,13; 12,13,14,15,16,17; 16,17,18,19,20,21;20,21,22,
23,24,25; 24,25,26,27,28,29; 28,29,30,31,32,1];
S1=[14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5, 9, 0, 7; 0, 15, 7, 4, 14, 2, 13, 1, 10, 6, 12, 11, 9, 5, 3, 8;
 4, 1, 14, 8, 13, 6, 2, 11, 15, 12, 9, 7, 3, 10, 5, 0; 15, 12, 8, 2, 4, 9, 1, 7, 5, 11, 3, 14, 10, 0, 6, 13];
```

```
S2=[15, 1, 8, 14, 6, 11, 3, 4, 9, 7, 2, 13, 12, 0, 5, 10; 3, 13, 4, 7, 15, 2, 8, 14, 12, 0, 1, 10, 6, 9, 11, 5;
0, 14, 7, 11, 10, 4, 13, 1, 5, 8, 12, 6, 9, 3, 2, 15; 13, 8, 10, 1, 3, 15, 4, 2, 11, 6, 7, 12, 0, 5, 14, 9];
S3=[10, 0, 9, 14, 6, 3, 15, 5, 1, 13, 12, 7, 11, 4, 2, 8; 13, 7, 0, 9, 3, 4, 6, 10, 2, 8, 5, 14, 12, 11, 15, 1;
 13, 6, 4, 9, 8, 15, 3, 0, 11, 1, 2, 12, 5, 10, 14, 7; 1, 10, 13, 0, 6, 9, 8, 7, 4, 15, 14, 3, 11, 5, 2, 12];
S4=[7, 13, 14, 3, 0, 6, 9, 10, 1, 2, 8, 5, 11, 12, 4, 15; 13, 8, 11, 5, 6, 15, 0, 3, 4, 7, 2, 12, 1, 10, 14, 9;
 10, 6, 9, 0, 12, 11, 7, 13, 15, 1, 3, 14, 5, 2, 8, 4; 3, 15, 0, 6, 10, 1, 13, 8, 9, 4, 5, 11, 12, 7, 2, 14];
S5=[2, 12, 4, 1, 7, 10, 11, 6, 8, 5, 3, 15, 13, 0, 14, 9; 14, 11, 2, 12, 4, 7, 13, 1, 5, 0, 15, 10, 3, 9, 8, 6;
 4, 2, 1, 11, 10, 13, 7, 8, 15, 9, 12, 5, 6, 3, 0, 14; 11, 8, 12, 7, 1, 14, 2, 13, 6, 15, 0, 9, 10, 4, 5, 3];
S6=[12, 1, 10, 15, 9, 2, 6, 8, 0, 13, 3, 4, 14, 7, 5, 11; 10, 15, 4, 2, 7, 12, 9, 5, 6, 1, 13, 14, 0, 11, 3, 8;
 9, 14, 15, 5, 2, 8, 12, 3, 7, 0, 4, 10, 1, 13, 11, 6; 4, 3, 2, 12, 9, 5, 15, 10, 11, 14, 1, 7, 6, 0, 8, 13];
S7=[4, 11, 2, 14, 15, 0, 8, 13, 3, 12, 9, 7, 5, 10, 6, 1; 13, 0, 11, 7, 4, 9, 1, 10, 14, 3, 5, 12, 2, 15, 8, 6;
 1, 4, 11, 13, 12, 3, 7, 14, 10, 15, 6, 8, 0, 5, 9, 2; 6, 11, 13, 8, 1, 4, 10, 7, 9, 5, 0, 15, 14, 2, 3, 12];
S8=[13, 2, 8, 4, 6, 15, 11, 1, 10, 9, 3, 14, 5, 0, 12, 7; 1, 15, 13, 8, 10, 3, 7, 4, 12, 5, 6, 11, 0, 14, 9, 2;
 7, 11, 4, 1, 9, 12, 14, 2, 0, 6, 10, 13, 15, 3, 5, 8; 2, 1, 14, 7, 4, 10, 8, 13, 15, 12, 9, 0, 3, 5, 6, 11];
P=[16,7,20,21,29,12,28,17,1,15,23,26,5,18,31,10,2,8,24,14,32,27,3,9,19,13,30,6,22,11,4,25];
PC1=[57,49,41,33,25,17,9,1,58,50,42,34,26,18,10,2,59,51,43,35,27,19,11,3,60,52,44,36,63,55,47,3
9,31,23,15,7,62,54,46,38,30,22,14,6,61,53,45,37,29,21,13,5,28,20,12,4];
PC2=[14,17,11,24,1,5,3,28,15,6,21,10,23,19,12,4,26,8,16,7,27,20,13,2,41,52,31,37,47,55,30,40,51,
45,33,48,44,49,39,56,34,53,46,42,50,36,29,32];
Ki=zeros(16,48); % Podključevi (eng.subkeys) od 48-bit dužine za 16 rundi
K_PC1=K(PC1);
C0=K_PC1(1:28); % Dijeli se ključna dva podključa po 28 bita
D0=K_PC1(29:56); % tj. Ci i Di podključevi
for i=1:16
if i==1||i==2||i==9||i==14||i==16
C0=[C0(2:end),C0(1)]; % Ovdje se vrši kružno pomijeranje ulijevo za jedno bitsko mjesto
D0=[D0(2:end),D0(1)]; % u 1. 2. 9. i 16. rundi
else
C0=[C0(3:end),C0(1:2)];% Ovdje se vrši bitsko kružno šiftovanje za dva bitska mjesta
 % ulijevou ostalim rundama: 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14 i 15
D0=[D0(3:end),D0(1:2)];
end
K_LS=[C0,D0];% Konkatenacija u novi podključ od 48 bita
Ki(i,:)=K_LS(PC2);% Potom permutacija tog podključa sa permutation choice 2 – PC2
% tako da se dobijuKi - podključevi rundi
end
L=M(1:32);% Ulazna 64-bitna poruka M (eng. message) se dijeli
% u dvije (L-left i R-right) poruke po 32 bita
R=M(33:64); % Ovo je desna 32-bitna poruka
for i=1:16% Pomoću ove for petlje se prolazi kroz 16-rundi
E0=reshape(E',1,48);
R_E=R(E0);% Vrši se ekspanizija i permutacija
R_Ki=mod(R_E + Ki(17-i,:),2);% XOR (sabiranje po modulu 2) sa podključem runde
B=R_Ki(1:6);% Potom, dolje se vrši S-box supstituciju 6 bitasa 4 odgovarajuća bita
x=B(1)*2+B(6)+1;
y=B(2)*8+B(3)*4+B(4)*2+B(5)+1;% Ovdje su: *8,*4,*2,*1 ustvari težine bitskih pozicija
C=['0000',dec2bin(S1(x,y))];% Ovdje se decimalna vrijednost konvertuje u binarnu vrijednost
C=C(end-3:end);% Poreda se u redosljed po LSB - Least Significant Bit, ne MSB
C1=[str2num(C(1)),str2num(C(2)),str2num(C(3)),str2num(C(4))];
B=R_Ki(7:12);
x=B(1)*2+B(6)+1;
y=B(2)*8+B(3)*4+B(4)*2+B(5)+1;
C=['0000',dec2bin(S2(x,y))];
C=C(end-3:end);
C2=[str2num(C(1)),str2num(C(2)),str2num(C(3)),str2num(C(4))];
B=R_Ki(13:18);
x=B(1)*2+B(6)+1;
y=B(2)*8+B(3)*4+B(4)*2+B(5)+1;
C=['0000',dec2bin(S3(x,y))];
C=C(end-3:end);
  C3=[str2num(C(1)),str2num(C(2)),str2num(C(3)),str2num(C(4))];
   B=R_Ki(19:24);
   x=B(1)*2+B(6)+1;
   y=B(2)*8+B(3)*4+B(4)*2+B(5)+1;
   C=['0000',dec2bin(S4(x,y))];
   C=C(end-3:end);
   C4=[str2num(C(1)),str2num(C(2)),str2num(C(3)),str2num(C(4))];
   B=R_Ki(25:30);
   x=B(1)*2+B(6)+1;
   y=B(2)*8+B(3)*4+B(4)*2+B(5)+1;
   C=['0000',dec2bin(S5(x,y))];
   C=C(end-3:end);
   C5=[str2num(C(1)),str2num(C(2)),str2num(C(3)),str2num(C(4))];
  B=R_Ki(31:36);
   x=B(1)*2+B(6)+1;
   y=B(2)*8+B(3)*4+B(4)*2+B(5)+1;
   C=['0000',dec2bin(S6(x,y))];
   C=C(end-3:end);
   C6=[str2num(C(1)),str2num(C(2)),str2num(C(3)),str2num(C(4))];
   B=R_Ki(37:42);
   x=B(1)*2+B(6)+1;
   y=B(2)*8+B(3)*4+B(4)*2+B(5)+1;
   C=['0000',dec2bin(S7(x,y))];
   C=C(end-3:end);
   C7=[str2num(C(1)),str2num(C(2)),str2num(C(3)),str2num(C(4))];
   B=R_Ki(43:48);
   x=B(1)*2+B(6)+1;
   y=B(2)*8+B(3)*4+B(4)*2+B(5)+1;
   C=['0000',dec2bin(S8(x,y))];
   C=C(end-3:end);
   C8=[str2num(C(1)),str2num(C(2)),str2num(C(3)),str2num(C(4))];
   C=[C1,C2,C3,C4,C5,C6,C7,C8];
   R_P=C(P);
   TEMP=L;
   L=R;
   R=mod(TEMP+R_P,2);
end
TEMP=L;
L=R;
R=TEMP;
C=[L,R];
CS=[];
C=num2str(C);
pos=find(C~=' ');
C=C(pos);
for i=1:4:61
  Ci=C(i:i+3);
  CS=[CS,num2str(dec2hex(bin2dec(Ci)))];
end
C=CS;
plain_text = [plain_text C];
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Listing 3.13: Implementacija ***myDES_Dec.m*** je kreirana na osnovu izvornih kdova u [9-10]

Četvrto zeleno dugme na **GUI app** sa slike 3 ima izvorni kd prikazan sljedećim listingom 3.14.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function steganography_using_random_AES_256_bit_key_for_each_data_block(app, event)
        clear all;close all; clc;tic;
        [y,Fs] = audioread('mySpeech.wav');
        sound(y,Fs)
        wavebinary = dec2bin( typecast( single(y), 'uint8'), 8 ) - '0';
        vidObj = VideoReader('Superman.mp4');
        vidHeight = vidObj.Height;
        vidWidth = vidObj.Width;
        mov = struct('cdata',zeros(vidHeight,vidWidth,3,'uint8'), 'colormap',[]);
        k = 1;
while hasFrame(vidObj)
        mov(k).cdata = readFrame(vidObj);
        k = k+1;
end
        In_hex = conversionTypeForAES(wavebinary);
 [OutChiperAES, all_KEYs] = chiperForAES_random_key(In_hex);% Šifrat
        OutWavebinary = conversionTypeForStego(OutChiperAES);
        OutputVideo = MojaEfikasnaSteganography( OutWavebinary, mov);
        snimi_u_avi_format_video(OutputVideo);
        RXvidObj = VideoReader('stegoSuperman.avi');
        vidHeight = RXvidObj.Height;
        vidWidth = RXvidObj.Width;
        r_mov = struct('cdata',zeros(vidHeight,vidWidth,3,'uint8'), 'colormap',[]);
        k = 1;
```

```
while hasFrame(RXvidObj)
        r_mov(k).cdata = readFrame(RXvidObj);
        k = k+1;
end
[y_format, Fs] = audioread('myFormatSpeech.wav');
 sound(y_format,Fs)
[wavdata, encAESwavebinary] = InverzMojaEfikasnaSteganography( r_mov, y_format );
sound(wavdata, Fs);
RX_In_hex = conversionTypeForAES(encAESwavebinary);
invOutChiperAES = invChiperForAES_random_key(RX_In_hex,all_KEYs);
rxOutWavebinary = conversionTypeForStego(invOutChiperAES);
encAES_wavdata = reshape( typecast( uint8(bin2dec( char(rxOutWavebinary + '0') )), 'single'
), size(y_format) );
sound(encAES_wavdata, Fs);
toc;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Listing 3.14: Izvorni kod 4. zelenog dugmeta kod **GUI app** sa slike 3

Izvorni MATLAB-ov kd za peto zeleno dugme sa slike 3 nećemo ovdje predstaviti i prepuštamo zainteresovanim čitaocima da to sami implementiraju. U tu svrhu, od pomoći je da spomenemo da se digitalno potpisivanje sa **RSA** (**R**ivest-**S**hamir-**A**dleman)[2] algoritmom, u ovom slučaju veoma jednostavno može realizovali modifikacijom i prilagođenjem već javno dostupne MATLAB-ove RSA implementacije [11].

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
>> MSE_full = zeros(1,436); % Inicijalno MSE_full je vektor vrsta od 436 nula
for m = 1:436                % U našem video zapisu smo imali ukupno 436 frejmova
Original_x = frame2im(mov(1,m)); % Pretvorimo frejm u sliku (image) da bi koristili imshow
x = frame2im(OutputVideo(1,m)); % OutputVideo je stego-video, a mov je originalni video
for i = 1:3     % Imamo RGB video zapis, tj. 3 boje: R (red), G (green) i B (blue)
for j = 1:480   %  Kod nas je veličina video zapisa bila VidHeight = 480 i
for l = 1:640   % vidWidth = 640 piksela.
MSE_full(1,m) = MSE_full(1,m) + (Original_x(j,l,i) - x(j,l,i))^2;%Zbir kvadrata apsolutne greške
end
end
end
MSE_full(1,m) = MSE_full(1,m)/(vidHeight*vidWidth*3);% Izračunavamo srednjekvadratnu
end% grešku MSE_full kod m-tog frejma, m = 1, 2, ..., 436.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Listing 3.15: Izračunavanje *MSE* u MATLAB-ovom komandom prozoru

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
>> NCC_full = zeros(1,436);% Vektor vrsta od NCC za 436 frejmova je inicijalno jednak 0
>>  for m = 1:436% Prolazimo kroz sve frejmove video zapisa, kod nas je bilo 436 frejmova
Original_x = frame2im(mov(1,m));% Pretvaramo frejm u image da bi mogli prikazati sa imshow
x = frame2im(OutputVideo(1,m)); % OutputVideo je stego-video zapis, a mov je originalni
for i = 1:3
for j = 1:480
for l = 1:640
numerator(1,m) = numerator(1,m) + Original_x(j,l,i)* x(j,l,i);% Brojilac – eng. numerator
denumerator(1,m) = denumerator(1,m) + Original_x(j,l,i)^2;% Imenilac – eng. denumerator
end
end
end
NCC_full(1,m) = numerator/denumerator;% NCC pojedinačnog m-tog frejma video zapisa
end
>> format long% postavimo korištenje long formata jer će inače pisati da je uvijek NCC = 1.0
>> min(NCC_full)% Prikažemo minimalnu vrijednost koja je skoro 1.0, a idealno je 1!
ans =
   0.999999999999999
>> max(NCC_full)
ans =
   1.000000000000002% Posljednja cifra može biti računska greška i trebalo bi je zanemariti ...
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Listing 3.16: Izračunavanje normalizovane kroskorelacije – *NCC*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
>> NCC_frame = 0;
>> for m = 1:436
Original_x = frame2im(mov(1,m));
x = frame2im(OutputVideo(1,m));
for i = 1:3
for j = 1:480
for l = 1:640
numerator(1,i) = numerator(1,i) + Original_x(j,l,i)* x(j,l,i);
denumerator(1,i) = denumerator(1,i) + Original_x(j,l,i)^2;
end
end
NCC(1,i)= numerator(1,i)/denumerator(1,i);
NCC_frame = NCC_frame + NCC(1,i);
end
NCC_uk(1,m) = NCC_frame/3;
NCC_frame = 0;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Listing 3.17: Redosljed naredbi za izračunavanje NCC_{RGB}

4. OPIS SIMULACIJA I REZULTATI TESTIRANJA

Izvorni (tzv. *plaintext poruka*) audio zapis pod nazivom **mySpeech.wav** veličine 128 kB i vremenskog trajanja od 1,5 sekundi je kreiran pomoću *SnimiFormatSpeech.m*. Nadalje, korišten je originalni video **Superman.mp4** (veličine 14,1 MB i trajanja od 24 sekunde) u koji se pomoću funkcije ***NasaSteganografija.m*** mijenjaju originalni biti najmanje težine kod bajta vrijednosti piksela od svake komponente boje (crvene, zelene, plave – *eng.* **RGB** (**R**ed, **G**reen, **B**lue) kod frejmova originalnog video zapisa, tj. implementirali smo takav oblik video LSB steganografije. Nadalje, novodobijeni stego-video zapis memorišemo u ***.avi*** format pomoću funkcije ***snimi_u_avi_format_video.m*** koja kao izlaz snima u radnom direktorijumu video zapis ***stegoSuperman.avi*** (veličine 383 MB i trajanja od 24 sekunde). Naravno, ne možemo u ovom slučaju izvršiti snimanje u neki od formata sa gubicima (kao što je format ***.mp4***) jer bi došlo do *„nepovratnog gubitka pojedinih utisnutih bita informacije te ne bi imali nikakve koristi od takvog video zapisa"*. Dakle, prilikom prenosa trebaće nam mnogo više vremena za prenos ili veći kapacitet prenosnog linka jer moramo koristiti nekompresovane *.avi* formate video zapisa, a i njihova veličina u zauzeću memorije će biti mnogo veća, a to ujedno predstavlja neke od nedostataka implementirane steganografije. Naravno, implementacija se može modifikovati da se koristi i kod standardnih kompresovanih formata video zapisa ukoliko bi znali tačno kao su implementirani. Međutim, ti standardi su danas zaštićeni patentima i to treba imati u vidu prilikom takvih opcionalnih unapređenja i modifikacija algoritama da ne bi prekršili zaštićena prava. Takođe, ovdje je veoma važno zapaziti jednu od mana steganografije koja koristi nekompresovani video format – tj. negativna potreba za mnogo većim „zaglavljem podataka" prilikom prenosa jer je originalni video

2 Skraćenica **RSA** potiče od naziva autora Ron **R**ivest, Adi **S**hamir i Lonard **A**dleman koji su prvi javno objasnili ovaj algoritam 1977. godine.

bio veličine 14,1 MB dok je stego-video zauzimao 383 MB memorije u radnom direktorijumu. Vizuelna predstava o subjektivnom utisku kvaliteta prenošenih stego-video frejmova i originalnih frejmova prikazana je na primjeru 1. frejma u nastavku na slici 4, odnosno slici 5 – respektivno. Sa ovih slika se zaključuje da utiskivanje audio bita u originalni video primjenom LSB steganografije ne dovodi do subjektivne mogućnosti da se razazna razlika između stego-video zapisa i originalnog video zapisa, a u ovom slučaju kod 1. frejma (ovo je ujedno bila maksimalna vrijednost u našem primjeru testiranog audia i videa gdje smo imali ukupno 436 frejmova u video zapisu) srednje-kvadratna greška je iznosila $MSE = 2{,}7670\times10^{-4}$ što je i objektivno zanemarljiva vrijednost. Naš redosljed komandi u komandnom prozoru MATLAB-a i opis fukcija korištenih za izračunavanje *MSE* grešaka kod svih frejmova smještenih unutar vektora *MSE_full*, prikazan je listingom 3.15.

Nadalje, izračunali smo *PSNR* koristeći sljedeću opštepoznatu formulu

$$PSNR = 10\log\left(\frac{C_{max}^2}{MSE}\right), \qquad (3)$$

gdje je C_{max} – maksimalna vrijednost piksela u originalnim frejmovima video zapisa. U našem slučaju (kada se pretvori u tip ***uint8***) ta vrijednost je iznosila očekivanih 255 (tj. 2^8-1).

Slika 4: Prikaz 1. frejma u stego-video zapisu (*stegoSuperman.avi*)

Slika 5: Prikaz 1. frejma u originalnom video zapisu bez steganografije

Uvrštavanjem *MSE* i C_{max} u izraz (3) dobijamo u našem slučaju $PSNR = 10\log\left(\frac{C_{max}^2}{MSE}\right) = 10\log\left(\frac{255^2}{2{,}7670\times10^{-4}}\right) \approx 83{,}7\,\text{dB}$ *što je praktično izvrstan kvalitet – odnosno veoma mala distorzija originalnih* video frejmova. Nadalje, izračunali smo *NCC* između originalnih i stego-video frejmova pomoću listinga 3.16 gdje je vektor *NCC_full* predstavljao vektor vrstu od *NCC* za svih 436 frejmova, tj. *NCC* kod pojedinačnog RGB video frejma smo izračunali korišćenjem sljedeće formule

$$NCC = \frac{\sum_{i=1}^{numCol}\sum_{j=1}^{vidHeight}\sum_{l=1}^{vidWidth} I_s(j,l,i)\times I_c(j,l,i)}{\sum_{i=1}^{numCol}\sum_{j=1}^{vidHeight}\sum_{l=1}^{vidWidth} I_c^2(j,l,i)}, \qquad (4)$$

pri čemu je: *numCol* =3 jer smo imali RGB video frejmove koji imaju tri boje, dimenzije frejmova (visina i širina u pikselima) su *vidHeight* = 480, *vidWidth* = 640, I_s – je oznaka za stego-sliku (*eng.* image) u boji dobijenu konverzijom frejma u image sa ugrađenom MATLAB – ovom funkcijom ***im2frame*** i sa I_c – označena je originalna slika dobijena konverzijom *(j,l,i)* – tog frejma originalnog video zapisa. Na kraju, kao što se vidi u listingu 3.16, u našem primjeru je implementacijom izraza (4) dobijeno $NCC \approx 1.0$ *što je odlična vrijednost.* U radu [4] je navedeno da se kod RGB slika definiše *kroskorelacija* NCC_{RGB} kao srednja vrijednost pojedinačnih kroskorelacija pomoću formule

$$NCC_{RGB} = \frac{\left[NCC_R + NCC_G + NCC_B\right]}{3}, \qquad (5)$$

gdje su: NCC_R – normalizovana kroskorelacija kod *R* – frejma, NCC_G – normalizovana kroskorelacija kod *G* – frejma i NCC_B – normalizovana kroskorelacija kod *B* – frejma jedne RGB slike. Primjenom izraza (5) izračunali smo $NCC_{RGB} = 1$ kod svih frejmova video zapisa, tj. kod *m* – tog frejma *NCC_uk(1,m)* = 1, *m* = 1, 2, ..., 436. Naravno, da bi se mogao izvršiti spisak komandi u listingu 3.16 i/ili listingu 3.17 prethodno je potrebno u komandnom prozoru MATLAB-a pokrenuti sve komande izvornog kda prve opcije (zelenog dugmeta) sa slike 4, tj. funkcije *plain_text_steganography.m*.

Poslije toga, uporedili smo vremena izvršavanja pojedinih kriptografskih opcija, odnosno redom smo pokretali kd svih pet opcija sa slike 3. Dobijena vremena izvršavanja mjerena između *tic* i *toc* kod *i* – te kriptografske opcije su označena sa *čije su vrijednosti zaokružene na sekunde i prikazane* u Tabeli 1.

Tabela 1: - Vremena izvršavanje pojedinih opcija sa slike 3

$T_1^{tic\to toc}[s]$	$T_2^{tic\to toc}[s]$	$T_3^{tic\to toc}[s]$	$T_4^{tic\to toc}[s]$	$T_5^{tic\to toc}[s]$
23,0	862,0	2049,0	927,0	887,0

Nadalje, dobijeni rezultati trajanja simulacija u Tabeli 1 su čisto informativnog karaktera jer kada se ponavljaju simulacije dolazi do varijanse u dobijenim rezultatima i ta vremena više služe da se ima osjećaj za red veličine trajanja vremena izvršavanja pojedinih kriptografskih opcija. Važno je zapaziti da je najduže trajala simulacija pod 3. rednim brojem (**Steganography with DES using 56-bit key for encryption/decryption of source audio record hidden in video movie**), tj. kada se uz steganografiju implementirala i DES enkripcija/dekripcija. Prva opcija steganografije sa slike 3 (**Plain_text LSB steganography for transmission of audio record hidden into video record**) imala vrijeme izvršavanja od zanemarljivih 23,0 sekundi u odnosu na ostalo vrijeme koje je bilo potrebno za izvršavanje drugih opcija sa slike 3 gdje su pridodate im-

plementacije DES i/ili AES enkripcije/dekripcije nad izvornim audio zapisom koji se naknadno utiskivao u video primjenom 1. opcije steganografije. Konačno, iako se simetrični kriptografski algoritmi brže izvršavaju u odnosu na asimetrične algoritme (tzv. algoritme sa javnim ključem) neoprezan čitalac bi mogao zaključiti na osnovu rezultata iz Tabele 1 da je 5. kriptografska opcija bila najbrža od korišćenih implementacija i to jeste tačno na osnovu tih rezultata, ali samo zbog toga što smo *u našem slučaju* koristili „pokaznu" RSA implementaciju sa relativno malim prostim brojevima za enkripciju, dekripciju, javni i tajni ključ; dok se u praksi RSA implementira sa mnogo većim prostim brojevima za te vrijednosti i tada sam postupak znatno duže traje.

5. ZAKLJUČAK

Od kada postoje prva pisma u civilizacijama javljala se i potreba za tajnim pisanjem i saopštavanjem informacija samo pogodnim osobama. U ovom radu smo predstavili pregled osnovnih informacija, kao i gdje je moguće pronaći ostale informacije ukoliko se javi potreba da neko implementira video steganografiju. Metode i implementacije u izvornom MATLAB-ovom kodu mogu biti od velike koristi u edukativne svrhe, što je nama prvenstveno i bio cilj da postignemo sa ovim člankom. Takođe, maštovitiji čitalac može vrlo lako iskoristiti postojeće izvorne fajlove radi neke nove implementacije steganografske varijacije u odnosu na ovdje implementirane opcije. Naravno, same se nameću i moguće kontra-mjere koje kritpoanalitičar može primjeniti ako se posumnja na primjenu ove, u nekim zemljama nedozvoljene, LSB steganografije. U tom smislu, otvara se opcija da se u prenosu slučajnim postupkom generišu i izmjene pravi LSB biti u poslatom stego-video zapisu čime bi se onemogućio prijem originalnog audio zapisa. Naravno, kriptografske opcije koje se mogu iskoristi su neiscrpne kod darovitog i vještog kriptografa, tako da svaka kontra-mjera može povući novu implementiranu zaštitnu/sigurnosnu mjeru. U budućem radu bi se moglo fokusirati na implementaciju dodatnih kriptografskih opcija kao što je LSB-BMSE metod predložen u [13]. Takođe, radi povećanja brzine izvršavanja implementiranih funkcija u MATLAB-u u daljnjem istraživanju bi sigurno trebalo staviti fokus na implementaciju sa FPGA jer je pokazano da je FPGA implementacija enkripcije (dekripcije) brža 696000 (236000) puta od implementacije u MATLAB-u [14]. Na kraju, radi budućih istraživanja bilo bi korisno implementirati steganografiju baziranu na adaptivnoj neuraloj mreži, ili nešto slično primjenom još heš (*eng.* hash) funkcija kao u radu [15]. Naravno, pri tom bi se u budućem istraživanju trebalo fokusirati i na moguću implementaciju steganografije u nekim kompresovanim i često korištenim video formatima kao u [16].

LITERATURA

[1] M. I. Mihailescu, S. L.Nita, *Cryptography and Cryptanalysis in MATLAB: Creating and Programming Advanced Algorithms*, New York, NY, USA: Apress, 2021.

[2] A. Fatnassi, H. Gharsellaoui, S. Bouamama, Towards Novel Video Steganography Approach for Information Security, Procedia Computer Science, Volume 159, 2019, Pages 953-962, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2019.09.262. [Online]. Available: (https://www.sciencedirect.com/science/article/pii/S1877050919314589). [Accessed: 25-Oct-2022].

[3] M. Hacimurtazaoglu, T. Kemal, "LSB-based pre-embedding video steganography with rotating & shifting poly-pattern block matrix." PeerJ. Computer science vol. 8 e843. 6 Jan. 2022, doi:10.7717/peerj-cs.843

[4] E. Emad, A. Safey, A. Refaat, Z. Osama, E. Sayed and E. Mohamed, "A secure image steganography algorithm based on least significant bit and integer wavelet transform," in Journal of Systems Engineering and Electronics, vol. 29, no. 3, pp. 639-649, June 2018, doi: 10.21629/JSEE.2018.03.21.

[5] Younus, Zeyad Safaa and Younus, Ghada Thanoon. "Video Steganography Using Knight Tour Algorithm and LSB Method for Encrypted Data" *Journal of Intelligent Systems*, vol. 29, no. 1, 2020, pp. 1216-1225. [Online]. Available: https://doi.org/10.1515/jisys-2018-0225. [Accessed: 05-Dec-2022]

[6] National Institute of Standards and Technology, "Data Encryption Standard (DES)," *CSRC*, 25-Oct-1999. [Online]. Available: https://csrc.nist.gov/publications/detail/fips/46/3/archive/1999-10-25. [Accessed: 04-Dec-2022].

[7] W. Stallings, *Cryptography and Network Security Principles and Practices*, 4th Edition, New Jersey, NJ, USA: Prentice Hall, Pearson Education Inc., 2005.

[8] National Institute of Standards and Technology, "Announcing the advanced encryption standard (AES) - NIST," 26-Nov-2001. [Online]. Available: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=901427. [Accessed: 05-Dec-2022].

[9] D. Hill, "Advanced encryption standard (AES)-128,192, 256," *MathWorks*, 25-Jan-2021. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/73412-advanced-encryption-standard-aes-128-192-256. [Accessed: 05-Dec-2022].

[10] "Freesourcecode.net," *Data encryption standard (des) in matlab | download free open source Matlab toolbox, matlab code, matlab source code*. [Online]. Available: http://freesourcecode.net/matlabprojects/59871/data-encryption-standard-%28des%29-in-matlab#.Y44iM3bMI2x. [Accessed: 20-Oct-2022].

[11] N. Felix, "Des(STR,key,mode)," *MathWorks*, 20-Jun-2019. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/53768-des-str-key-mode?s_tid=srchtitle. [Accessed: 19-Oct-2022].

[12] V. Wilms, "RSA public key encryption and signing (32bit)," *MathWorks*, 11-Oct-2015. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/53457-rsa-public-key-encryption-and-signing-32bit?s_tid=prof_contriblnk. [Accessed: 25-Oct-2022].

[13] M. M. Mahmoud and H. T. Elshoush, "Enhancing LSB Using Binary Message Size Encoding for High Capacity, Transparent and Secure Audio Steganography–An Innovative Approach," in IEEE Access, vol. 10, pp. 29954-29971, 2022, doi: 10.1109/ACCESS.2022.3155146.

[14] B. K. Yakti, S. Madenda, S. A. Sudiro and P. Musa, "Processing Speed Comparison of the Least Significant Bit (LSB) Steganography Algorithm on FPGA and Matlab," 2021 Sixth International Conference on Informatics and Computing (ICIC), 2021, pp. 1-7, doi: 10.1109/ICIC54025.2021.9632978.

[15] K. J. Velmurugan and S. Hemavathi, "Video Steganography by Neural Networks Using Hash Function," 2019 Fifth International Conference on Science Technology Engineering and Mathematics (ICONSTEM), 2019, pp. 55-58, doi: 10.1109/ICONSTEM.2019.8918877.

[16] I. Almomani, A. Alkhayer, and W. El-Shafai, "A Crypto-Steganography Approach for Hiding Ransomware within HEVC Streams in Android IoT Devices," *Sensors*, vol. 22, no. 6, p. 2281, Mar. 2022, doi: 10.3390/s22062281. [Online]. Available: http://dx.doi.org/10.3390/s22062281. [Accessed: 05-Dec-2022].

Boris Malčić, ma, Elektrotehnički fakultet, Univerzitet u Banjoj Luci, RS, BiH.
Kontakt: boris.malcic@etf.unibl.org
Oblast interesovanja: Radio komunikacije, digitalne telekomunikacije, elektronika, mikrotalasna tehnika, antenski sistemi, 5G/6G mobilne mreže, radarski sistemi, lokalizacija u radio mrežama, prenos i obrada multimedijalnih signala.

Prof. dr Zoran Đurić, Elektrotehnički fakultet, Univerzitet u Banjoj Luci, RS, BiH.
Kontakt: zoran.djuric@etf.unibl.org
Oblast interesovanja: sigurnost, kriptografija, PKI, platni sistemi i protokoli, formalna verifikacija, mašinsko učenje, data science, objektno-orjentisano programiranje i modelovanje, Internet programiranje, razvoj mobilnih aplikacija, XML-bazirana međuoperativnost, Web servisi, računarske mreže, penetration testing, sistem integracija

Prof. dr Slavko Šajić, Elektrotehnički fakultet, Univerzitet u Banjoj Luci, RS, BiH.
Kontakt: slavko.sajic@etf.unibl.org
Oblast interesovanja: Agilni radio-komunikacioni sistemi, prenos signala u proširenom spektru, problemi i realizacija sinhronizacije u radio komunikacijama, digitalne telekomunikacije, RF elektronika, antenski i radarski sistemi, obrada i efikasan prenos multimedijalnih signala, lokalizacija i satelitski sistemi.

infoM

UPUTSTVO ZA PRIPREMU RADA

1. Tekst pripremiti kao Word dokument, A4, u kodnom rasporedu 1250 latinica ili 1251 ćirilica, na srpskom jeziku, bez slika. Preporučen obim – oko 10 strana, single prored, font 11.
2. Naslov, abstakt (100-250 reči) i ključne reči (3-10) dati na srpskom i engleskom jeziku.
3. Jedino formatiranje teksta je normal, bold, italic i bolditalic, VELIKA i mala slova (tekst se naknadno prelama).
4. Mesta gde treba ubaciti slike, naglasiti u tekstu (Slika1...)
5. Slike pripremiti odvojeno, VAN teksta, imenovati ih kao u tekstu, radi identifikacije, u sledećim formatima: rasterske slike: jpg, tif, psd, u rezoluciji 300 dpi 1:1 (fotografije, ekranski prikazi i sl.), vektorske slike – cdr, ai, fh,eps (šeme i grafikoni).
6. Autor(i) treba da obavezno priloži svoju fotografiju (jpg oko 50 Kb), navede instituciju u kojoj radi, kontakt i 2-4 oblasti kojima se bavi.
7. Maksimalni broj autora po jednom radu je 5.

Redakcija časopisa Info M