به نام خدا

CA 3

امیرعلی شهریاری (۸۱۰۱۰۰۱۷۳)

۱:

در این فاز از پروژه می خواهیم یک متن انگلیسی را در یک تصویر سیاه و سفید رمزنگاری کنیم . با توجه به روش های مختلفی که در کلاس بررسی و تحلیل کردیم و راه های مختلفی مثل اینکه هر چند بیت رو کامل برای رمزنگاری تصویر خواسته شده قرار دهیم که با بررسی مشکلات آن نظیر افشا شدن آن یا بهم ریختن تصویر و خراب شدن بخش هایی از آن ، در نهایت به روشی رسیدیم که بیت کم ارزش هر پیکسل از تصویر را برای رمز نگاری انتخاب کنیم که با انجام آن کمترین تغییر در تصویر مشهود خواهد بود و میتوان تصویر را رمزنگاری کرد بدون آنکه از ظواهر تصویر این اتفاق پیدا باشد.

۱-۱:

در این بخش میخواهیم یک مپ ست ۳۲ عضوی از حروف و علایم انگلیسی که میخواهیم در پیاممان از آن استفاده کنیم ، ایجاد کنیم.

```
number = 32;
dec2bin(0:31,5);
mapset = cell(2, number);
alphabet = 'abcdefghijklmnopqrstuvwxyz .,!";';

for i = 1:number
    mapset{1,i} = alphabet(i);
    mapset{2,i} = dec2bin(i-1, 5);
end
```

همانطور که در کد مشخص است با کمک دستور dec2bin نخست تعداد ۳۲ کارکتر رو مشخص کرده و ۵ را به عنوان ۵ بیتی که میخواهیم به کمک آن مپ ست را تشکیل دهیم مشخص میکنیم. سپس الفبای رمزنگاری مان که شامل حروف کوچک انگلیسی ، اسپیس ، علامت تعجب و ... است را مشخص و در حلقه آنها را به هر باینری متناظر نسبت می دهیم.

در نهایت مپ ست سلولی ۲*۳۲ به شکل فوق تشکیل شده که به هر کدام ۵ بیت متناظر را مرتبط کرده:

2x32 cell

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	a	b	c	d	e	f	g	h	i	j	k	l	m	n
2	00000	00001	00010	00011	00100	00101	00110	00111	01000	01001	01010	01011	01100	01101

2x32 cell

	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
1		s	t	u	v	w	x	y	z		.	,	!	"	;
2	)1	10010	10011	10100	10101	10110	10111	11000	11001	11010	11011	11100	11101	11110	11111

۲-۱:

```
image = imread("picture.png");
image = rgb2gray(image);

message='signal;';
message_len=length(message);
message_binary=cell(1,message_len);

for i = 1:message_len
    current_char=message(i);
    index=strcmp(current_char,mapset(1,:));
    message_binary(i) = mapset(2,index);
end


binarymessage_encoded=cell2mat(message_binary);
binarymessage_len=length(binarymessage_encoded);
encoded_image =image;

for i = 1:binarymessage_len
    vals=image(i);
    valsbin1=dec2bin(vals);
    valsbin1(end)=binarymessage_encoded(i);
    encoded_image (i)=bin2dec(valsbin1);
end
encrypted_filename = 'encrypted_image.png';
imwrite(encoded_image , encrypted_filename);

subplot(1,2,1);
imshow(image);
title("Original photo")
subplot(1,2,2);
imshow(encoded_image);
title("Encrypted photo")
```

به مانند کدی که در کلاس با توجه به راه حل پیشنهادی تدوین کردیم ، نخست تصویر دلخواه را با کمک دستور rgb2gray به یک تصویر سیاه و سفید تبدیل کرده و در ادامه متن پیغام را که میخواهیم رمزنگاری کنیم نوشته و برای نشان دادن اتمام آن ، از (;) استفاده می کنیم. در نخستین for کد ، پیغام را نیز تبدیل به رشته باینری کرده و با توجه به آنچه در مپ ست پیشتر تعریف کردیم ، آنرا مپ کرده و در message_binary ذخیره می کنیم. به کمک cell2mat یک رشته باینری طویل ایجاد شده .

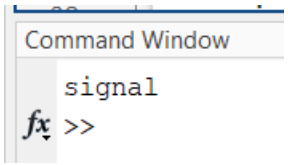
حالا encoded_image را کپی از تصویر اولیه نگه میداریم و در حلقه بعدی حال روی هر پیکسل تصویر پیمایش می کنیم و عملا مقدار باینری پیکسل تصویر را استخراج میکنیم . در valsbin بیت آخری پیام را گرفته و بیت i جیگزین میگردد سپس مقدار باینری بروز شده به دسیمال تبدیل گشته و پیکسل i آنرا با مقدار جدید تنظیم میکنیم. و در نهایت تصویر را به صورت png سیو کرده و همچنین دو تصویر را نیز ساب پلات می کنیم.

۳-۱: در تصویر بالا هردو شکل نمایان است و با کنار هم قرار دادن آنها ، با توجه به توضیحات اول پروژه و تاثیر کمی که تغییر کم ارزش ترین پیکسل دارد ، عملا تفاوت قابل رویتی را نمیتوان دید و دو تصویر مانند یکدیگر هستند.

۴-۱:

```
encrypted_filename = 'encrypted_image.png';
encrypted_photo = imread(encrypted_filename);


message_bin = '';
for i = 1:size(encrypted_photo,1)
    pixel = encrypted_photo(i);
    pixel_bin = dec2bin(pixel);
    message_bin = [message_bin pixel_bin(end)];
end

message_decoded = '';
while length(message_bin) > 4
    char_bin = message_bin(1:5);
    message_bin = message_bin(6:end);

    index = find(strcmp(char_bin, mapset(2,:)));
    char = mapset{1,index};

    if char == ';'
        break;
    end
    message_decoded = [message_decoded char];
end

disp(message_decoded)
```

نخست تصویری که در بخش قبلی رمزنگاری کردیم را به عنوان ورودی گرفته و یک بلنک باینری تعریف میکنیم تا عملا نگهداری کنه از مقادیر و با کمک حلقه فور روی پیکسل ها پیمایش انجام می دهیم و ۸ بیت آن را اکسترکت کرده و و به مانند عکس آنچه در انکودر انجام دادیم بیت آخر آنرا برای پیاممان جدا می کنیم و پیام کامل کد گذاری شده در باینری ساخته می شود . حالا یک رشته خالی برای نگهداری کارکتر های رمزگشایی شده میسازیم و ۵ بیت اول رو برداشته و برای نگاشت آن با مپ ستی که ساختیم استفاده میکنیم و رشته را به رشته پیام رمزگشایی شده اضافه می کنیم و سپس حذف میکنیم آن ۵ بیت را و در بخش شرطی هرگاه به ; که بیانگر پایان پیام است رسیدیم ، عملیات متوقف می شود و در نهایت خروجی رمزگشایی شده چاپ می شود.



۵-۱: شاید تا حدودی بستگی به میزان نویز و اختلال وارده داشته باشد و احتمالا برخی از پیکسل هایمان کاملا تغییر می کنند و بنابراین باعث ایجاد خطا در کارکتر های ۵ بیتی رمزگشایی شده میشود . در حالت های خفیف تر ممکنه بعضی از حروف تغییر کنند که با توجه به افزونگی زبان میتوان آنرا با حرف اصلی جایگزین کرد و نویز و مشکل پیام را رفع کرد . همچنین روش هایی چون ایجاد بیت های برابری در کل مقدار پیام قابل بازیابی به تعداد بیت های خراب بستگی دارد. اگر فقط بخش کوچکی از

پیکسل ها تحت تأثیر نویز قرار گیرند، ممکن است بیشتر پیام را دست نخورده دریافت کنیم یا با ویرایش کوچکی آنرا تصحیح کنیم ؛ اما یک سطح نویز بالاتر می تواند به طور بالقوه بیت های زیادی را برای بازیابی کامل پیام پنهان خراب کند.

۶-۱:روش های متعددی برای این امر وجود دارد ، روش هایی چون:

تجزیه و تحلیل هیستوگرام : بررسی میکنیم که آیا هیستوگرام صفحه LSB به طور یکنواخت توزیع شده است. تصاویر رمزگذاری نشده ساختار بیشتری خواهند داشت و داده های رمزگذاری شده تصادفی تر به نظر می رسند.

Chi-square test: این آزمایش تصادفی بودن، آمار Chi-squareرا در صفحه LSB محاسبه میکند. مقادیر p پایین نشان می دهد که LSB ها تصادفی ورمزگذاری شده تر هستند.

تجزیه و تحلیل آنتروپی:با محاسبه آنتروپی صفحه LSB داده های رمزگذاری شده دارای آنتروپی بالاتری نسبت به داده های رمزگذاری نشده خواهند بود.

تجزیه و تحلیل همبستگی : تصاویر رمزگذاری نشده همبستگی بالاتری بین پیکسل های همسایه دارند. این در تصاویر رمزگذاری شده کاهش می یابد.

یادگیری ماشینی : مدلی مانند روش‌های SVM، CNN یا مجموعه‌ای را آموزش دهید تا تصاویر رمزگذاری‌شده و رمزگذاری‌نشده را تشخیص میدهد که البته این نیازمند دیتا ست خوبی از پیش است که بتوانیم حالات رمز نگاری شده و نشده را به آن بدهیم تا یاد بگیرد و سپس با احتمالی تخمین بزند که تصویر پیش رو بیشتر شبیه کدام یک از حالات است.

۱-۲: در این بخش به کمک توضیحات و بخش راهنمایی آن می خواهیم صوت متناظر با شماره داده شده را بدست آورده و با کمک دستور sound پخش کنیم.

```
generate_dtmf_tones('43218765');

function generate_dtmf_tones(my_number)

    fr=[697 770 852 941];
    fc=[1209 1336 1477];
    fs=8000;
    Ts=1/fs;
    Ton=0.1;
    Toff=0.1;
    t=0:Ts:Ton;
    silence=zeros(1,size(t,2)-1);
    out=[];

    n = 1;
    while n <= length(my_number)
        if my_number(n) == '*'
            row=4;
            column=1;
        elseif my_number(n) == '#'
            row=4;
            column=3;
        elseif my_number(n) == '0'
            row=4;
            column=2;
        else
            num=str2num(my_number(n));
            row=ceil(num/3);
            column=rem(num,3);
            if column==0
                column=3;
            end
        end
        y1=sin(2*pi*fr(row)*t);
        y2=sin(2*pi*fc(column)*t);
        y=(y1+y2)/2;
        on=Ton*fs;
        out=[out y(1:on) silence];
        n = n + 1;
    end

    sound(out,fs)
    audiowrite('y.wav',out,fs)

end
```

بخش نخست کد به مانند بخش راهنمایی ، با استفاده از سیگنالینگ DTMF تولید کرده. مدت زمان هر سیگنال را ۰.۱ ثانیه و فاصله زمانی بین پخش دو سیگنال را هم ۰.۱ ثانیه در نظر بگیرید. فرکانس نمونه برداری را ۸ کیلوهرتز در نظر بگیرید.

سکوت میان کلید ها با تابعی از ۰ به طول t تعریف کرده ؛ در هر حلقه به ازای هر عدد سیگنالش تولید شده و پس از آن با یک سکوت ذخیره سازی میکنیم همچنین در این الگو #-*-0 خارج را نیز جدا کرده و به صورت شرطی آنها را تفکیک می کنیم و ردیف و ستون هر یک دستی مشخص می شوند. به این صورت که ردیف آن سقف آن بر ۳ بوده و ستون هر عدد باقی مانده آن بر ۳ است که چنانچه ۰ بود عملا همان ۳ و برای ۱ و ۲ خودشان قرار میگیرند ؛ این رویه ادامه دار است و همه را به ترتیب در out ذخیره کرده و در نهایت wave.out خروجی بدست آمده و با کمک دستور sound آنرا می شنویم.

۳-۲:

در این بخش که عملا به نوعی میخواهیم دیکود بخش قبلی را انجام داده و یک صوت را که به روش DTMF آنالیز کرده و بیان می کنیم که این سیگنال بر آمده از فشردن کدام کلید ها بوده است.

```
filename="a.wav";
[a,Fs]=audioread(filename);
sound(a,Fs)
data=cell(2,12);
fr=[697 770 852 941];
fc=[1209 1336 1477];
fs=8000;
Ts=1/fs;
Ton=0.1;
t=0:Ts:Ton;
Toff=0.1;
on=Ton*fs;
s=size(a);
data_name=['1','2','3','4','5','6','7','8','9','*','0','#'];
output=[];
for n=1:12
    num=n;
    row=ceil(num/3);
    column=rem(num,3);
    if column==0
        column=3;
    end
    y1=sin(2*pi*fr(row)*t);
    y2=sin(2*pi*fc(column)*t);
    y=(y1+y2)/2;
    data(1,n)={data_name(n)};
    data(2,n)={y(1:on)};
end

for n=0:(s(1)/(0.2*Fs))-1
    samples=[(2*n*Fs*0.1)+1,(2*n*Fs*0.1)+on];
    [b,Fs]=audioread(filename,samples);
    ro=zeros(1,12);
    for i=1:12
        ro(i)=corr2(data{2,i},transpose(b));
    end
    [MAXRO,pos]=max(ro);
    na=cell2mat(data(1,pos));
    output=[output na];
end
output
```

در اینجا نام کلید ها را در آرایه ای از کارکتر ها تعریف کرده و عملا به مشابه بخش قبلی پیاده سازی به ترتیب سیگنال هر کلید جنریت شده و و با نام آن در ردیف اول و دوم هر ستون سلول ۲ در ۱۲ ذخیره سازی می کنیم.

برای هر ورودی باید ۰.۱ ثانیه از آن را جدا کنیم و با اطلاع از طور فرکانس نمونه برداری شده الگویی برای محل هر سیگنال در فایل یافت می کنیم . شروع سیگنال n ام از 1+n*Fs*0.1*2 خواهد بود. اکنون با کورولیشن گیری و یافتم ماکسیمم، کلید مربوطه یا می یابیم سپس در حلقه بعدی و کورولیشن گیری همه سیگنال ها داخل سلول max یافت می شود و از بالای آن علامت کلید را استخراج کرده و در output ذخیره سازی می کنیم.

```
Command Window

output =

    '810198'
```

۴:

در این قسمت میخوایم با کمک template matching و کورولیشن تصویر یک ic را از مدار pcb استخراج کند و اطراف آنها مستطیل رسم کرده و آنرا جدا کند .

```
PCB_image = select_image();
IC_image = select_image();

PCB_image_gray = rgb_to_gray(PCB_image);
IC_image_gray = rgb_to_gray(IC_image);
IC_image_gray_rotated = imrotate(IC_image_gray,180);

M = corr_matrix(PCB_image_gray, IC_image_gray);
M_rotated = corr_matrix(PCB_image_gray, IC_image_gray_rotated);
M_cell = {M, M_rotated};

result = plot_box(PCB_image, IC_image_gray, M_cell);

figure();

subplot(1,3,1);
imshow(PCB_image);
title("PCB Image");

subplot(1,3,2);
imshow(IC_image);
title("IC Image");

subplot(1,3,3);
imshow(result);
title('Matching Result');

figure();
subplot(1,2,1);
imshow(IC_image);
title("IC Image");

function picture = select_image()
    [file, path] = uigetfile({'*.jpg;*.bmp;*.png;*.tif'}, 'choose');
    s = fullfile(path, file);
    picture = imread(s);
end
```

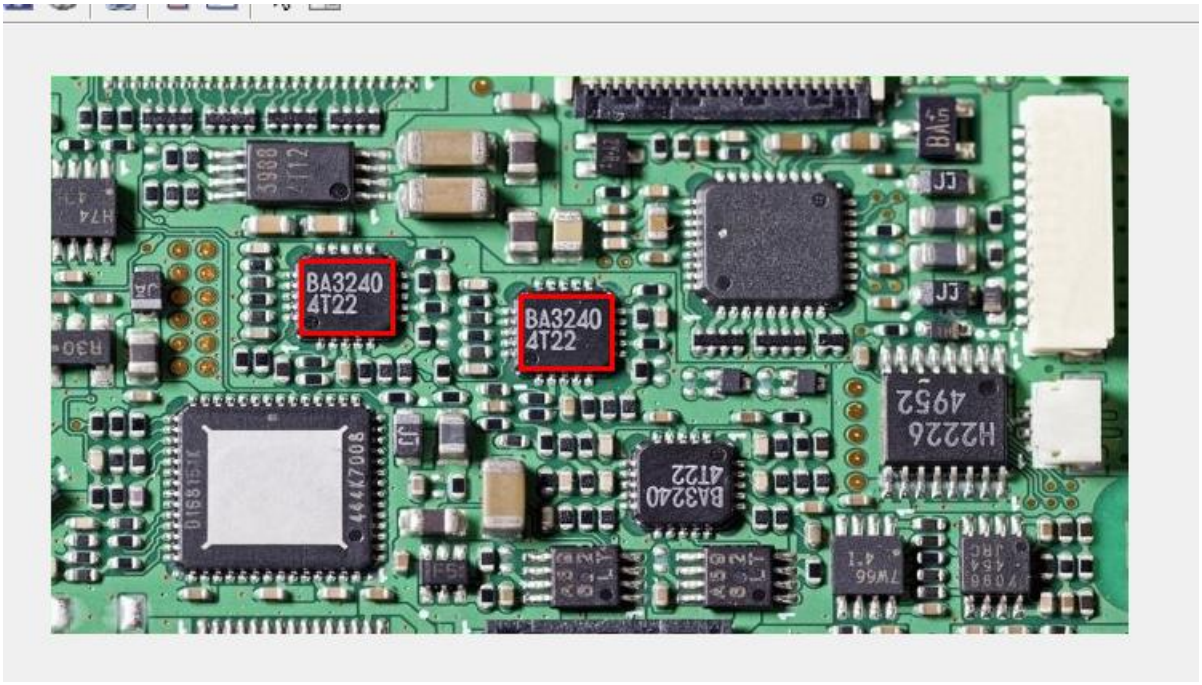
```
function grayscale = rgb_to_gray(picture)
    grayscale = 0.299 * picture(:,:,1) + 0.578 * picture(:,:,2) + 0.114 * picture(:,:,3);
end

function correlation_coef = corr_2d(picture1, picture2)
    a = sum(picture1(:) .* picture2(:));
    b = sqrt(sum(picture1(:).^2) .* sum(picture2(:).^2));
    correlation_coef = a / b;
end

function M = corr_matrix(PCB, IC)
    [PCB_row, PCB_col] = size(PCB);
    [IC_row, IC_col] = size(IC);
    IC = double(IC);
    M = zeros(PCB_row - IC_row + 1, PCB_col - IC_col + 1);
    for i = 1:(PCB_row - IC_row + 1)
        for j = 1:(PCB_col - IC_col + 1)
            PCB_cropped = double(PCB(i:i + IC_row - 1, j:j + IC_col - 1));
            M(i,j) = corr_2d(IC, PCB_cropped);
        end
    end
end

function result = plot_box(PCB, IC, M_cell)
    [IC_row, IC_col] = size(IC);
    threshold = 0.95;
    figure, imshow(PCB);
    hold on
    for l = 1:length(M_cell)
        M_1 = cell2mat(M_cell(1,l));
        [rows, columns] = find(M_1 > threshold);
        for k = 1:length(rows)
            rectangle('Position', [columns(k), rows(k), IC_col, IC_row], 'EdgeColor', 'r', 'LineWidth', 2);
        end
    end
    F = getframe(gcf);
    result = frame2im(F);
end
```

نخست در ورودی دو تصویر دو عکس pcb و ic را از کاربر دریافت کرده و سپس محتوای عکس را به ماتریسی سه بعدی تبدیل کرده و به تابع بر می گردانیم . همچنین مشابه بخش های قبلی برای سهولت کار آنرا به تصویر خاکستری تبدیل می کنیم . در ادامه به کمک تابع corr_2d دو تصویر را ورودی دریافت کرده و دو ماتریس تعریف می کنیم و ضریب همبستگی را محاسبه می کنیم و خروجی تابع باز می گرداند.

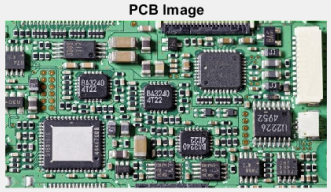
به کمک تابع corr_matrix در حلقه ای تصویر خاکستری قطعه را از گوشه بالا سمت چپ روی تصویر خاکستری برد مدار چاپی حرکت داده و با فراخواندن تابع d2_corr ضریب همبستگی بین تصویر قطعه و بخشی از تصویر برد مدار چاپی که در هر مرحله انتخاب کردیم را محاسبه میکنیم و مقدار آن را در ماتریس M ذخیره کرده. در انتها این ماتریس را به عنوان خروجی تابع قرار میدهیم. دستور imrotate تصویر ۱۸۰ درجه میچرخد و تصویر چرخانده شده و تصویر برد را فراخوانی کرده و ماتریس ضرایب خروجی قرار میدهیم مجددا تابع corr_matrix را فراخوانی کرده ، تصویر خاکستری بدون چرخش و تصویر خاکستری برد مدار چاپی را به آن میدهیم و خروجی آن را M قرار میدهیم M وM_rotate در سلولی به نامM_cell میریزیم.

اکنون تابع plot_box را تعریف کرده و تصویر رنگی برد مدار چاپی و تصویر خاکستری قطعه وM_cell را به عنوان ورودی دریافت میکند. سپس با تصویر رنگی برد مدار چاپی را نمایش می دهیم و با دستور hold onآن را نگه داشته تا بتوانیم مستطیل

را روی این تصویر رسم کنیم. سپس با نوشتن یک حلقه ابتدا سلول M را با دستور cell2mat به ماتریس تبدیل کرده و سپس مختصات عنصری از ماتریس M که بیشتر از حد آستانه است را با دستور find پیدا کرده و ذخیره میکنیم. سپس حلقه دیگری در حلقه قبلی نوشته و با دستور rectangleمستطیلی به ابعاد قطعه و در مکان مشخص شده که همان مکان قطعه در تصویر برد مدار چاپی است رسم میکنیم . با ران شدن قطعه کد اصلی توابع فراخوانی و ساب پلات کنار هم و تصویر IC را جداگانه خواهیم داشت.

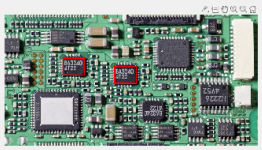


PCB Image



IC Image

Matching Result



IC Image