

TÜRKİYE CUMHURİYETİ  
YILDIZ TEKNİK ÜNİVERSİTESİ  
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ



RUBİK KÜPÜ PRENSİBİ İLE ŞİFRELENEN  
GÖRÜNTÜLERİN KRİPTOANALİZİ

17011076 – Ömer Buğrahan ÇALIŞKAN

17011039 – Ömer Aras KAPLAN

BİLGİSAYAR PROJESİ

Danışman  
Doç. Dr. Sırma YAVUZ

Haziran, 2021



## **TEŞEKKÜR**

---

Bize bu tür projeler yapma fırsatı sağlayan Yıldız Teknik Üniversitesi Bilgisayar Mühendisliği Bölümüne teşekkürlerimizi sunarız.

Lisans eğitimimde bilgisayar projesi dersi kapsamında beraber çalışma fırsatı bulduğumuz ve proje konumuzlarındaki bilgi ve tecrübesiyle bize katkıda bulunan değerli hocamız Sayın Doç.Dr. Sırma YAVUZ'a sonsuz teşekkürlerimizi sunarız.

Üniversite hayatımız boyunca bize kazandırdıkları her şey için diğer üniversite hocalarımıza da teker teker teşekkürlerimizi sunarız.

Ömer Buğrahan ÇALIŞKAN  
Ömer Aras KAPLAN

# İÇİNDEKİLER

---

<b>SİMGE LİSTESİ</b>	v
<b>KISALTMA LİSTESİ</b>	vi
<b>ŞEKİL LİSTESİ</b>	vii
<b>TABLO LİSTESİ</b>	ix
<b>ÖZET</b>	x
<b>ABSTRACT</b>	xi
<b>1 Giriş</b>	1
1.1 Şifreleme Nedir? . . . . .	1
1.2 Görüntü Şifreleme . . . . .	1
1.3 Rubik Küpü Görüntü Şifreleme Algoritması . . . . .	2
<b>2 Ön İnceleme</b>	3
2.1 Rübik'in Kübü Prensibine Dayanan Görüntü Şifreleme Algoritması . . . . .	3
2.2 Geliştirilmiş Hiperkaotik Dizilere Dayalı Görüntü Şifreleme Algoritması	3
2.3 Dairesel Pikseller Arası Bit Seviyesi Permütasyonu ile Kaos Tabanlı Görüntü Şifreleme Düzeninin Criptanaliz ve İyileştirilmesi . . . . .	4
2.4 Kaldırma Saldırılarının Sayısal Görüntü Filigranları Üzerindeki Etkisinin Uzamsal ve Frekans Alanı Analizi . . . . .	4
2.5 Görüntü Şifreleme Teknikleri Üzerine Bir Literatür İncelemesi . . . . .	5
2.6 L1-L2 TV Modelini Kullanarak Medyan Filtreleme için Geri Döndürme Tekniği . . . . .	5
<b>3 Fizibilite</b>	6
3.1 Teknik Fizibilite . . . . .	6
3.2 İş Gücü ve Zaman Planlanması . . . . .	7
3.3 Ekonomik Fizibilite . . . . .	7
<b>4 Sistem Analizi</b>	8

4.1	Hedefler . . . . .	8
4.2	Sistem Çalışması . . . . .	9
<b>5</b>	<b>Sistem Tasarımı</b>	<b>13</b>
5.1	Yazılım Süreci . . . . .	13
5.2	Program Kullanımı . . . . .	13
<b>6</b>	<b>Uygulama</b>	<b>15</b>
<b>7</b>	<b>Deneysel Sonuçlar</b>	<b>18</b>
7.1	Kaba-Kuvvet Saldırı . . . . .	18
7.2	Salt and Pepper Saldırısı . . . . .	21
7.3	Cropping Saldırısı . . . . .	24
<b>8</b>	<b>Performans Analizi</b>	<b>27</b>
8.1	Şifreleme - Deşifreleme Aşamaları . . . . .	27
8.2	Kaba-Kuvvet Saldırı Aşamaları . . . . .	28
8.3	Salt and Pepper Saldırı Aşamaları . . . . .	29
8.4	Cropping Saldırı Aşamaları . . . . .	30
<b>9</b>	<b>Sonuç</b>	<b>31</b>
	<b>Referanslar</b>	<b>32</b>
	<b>Özgeçmiş</b>	<b>33</b>

## SİMGELİSTESİ

---

$\oplus$  XOR

## KISALTMA LİSTESİ

---

XOR	Exclusive OR Logical Operation
IDE	Integrated Development Environment
KR	Satır Anahtarı (Key Row)
KC	Sütun Anahtarı (Key Column)
REVKR	Ters Çevirilmiş Satır Anahtarı (Reversed Key Row)
REVKC	Ters Çevirilmiş Sütun Anahtarı (Reversed Key Column)
SN	Saniye
KB	KiloByte
GHZ	Gigahertz

## ŞEKİL LİSTESİ

---

Şekil 3.1 Python Kütüphaneleri . . . . .	6
Şekil 3.2 Gantt Diyagramı . . . . .	7
Şekil 4.1 Image Dosyasının Okunması . . . . .	9
Şekil 4.2 KR ve KC Dizilerinin Oluşturulması . . . . .	9
Şekil 4.3 Kaydırma İşlemleri . . . . .	9
Şekil 4.4 Kaydırma Fonksiyonları . . . . .	10
Şekil 4.5 XOR Fonksiyonu . . . . .	10
Şekil 4.6 XOR İşlemleri . . . . .	10
Şekil 4.7 Deşifreleme İşlemleri . . . . .	11
Şekil 4.8 Algoritma Akış Şeması . . . . .	12
Şekil 5.1 Yazılım Akış Şeması . . . . .	13
Şekil 5.2 Program Kullanıcı Şeması . . . . .	14
Şekil 6.1 Lena Orijinal Görüntü . . . . .	15
Şekil 6.2 Şifrelenmiş Lena Görüntüsü . . . . .	15
Şekil 6.3 Deşifrelenmiş Lena Görüntüsü . . . . .	16
Şekil 6.4 Orijinal Lena - Şifrelenmiş Lena . . . . .	16
Şekil 6.5 Orijinal Lena - Deşifrelenmiş Lena . . . . .	16
Şekil 6.6 Program Terminal Çıktısı . . . . .	17
Şekil 7.1 Kaba-Kuvvet Saldırı Görüntü Tanımlamaları . . . . .	18
Şekil 7.2 Rastgele Üretilen KR ve KC dizileri . . . . .	19
Şekil 7.3 Kaba-Kuvvet Saldırı İşlemleri . . . . .	19
Şekil 7.4 Kaba-Kuvvet Saldırı Döngü Değişkeni . . . . .	20
Şekil 7.5 Kaba-Kuvvet Saldırı Başarı Durumu . . . . .	20
Şekil 7.6 Görüntüye Salt and Pepper Ekleme Fonksiyonu . . . . .	21
Şekil 7.7 Şifrelenmiş ve Salt and Pepper Eklenmiş Görüntü . . . . .	22
Şekil 7.8 Şifrelenmiş ve Salt and Pepper Eklenmiş Görüntülerin Benzerlik Grafiği . . . . .	22
Şekil 7.9 Deşifrelenmiş Orijinal ve Salt and Pepper Eklenmiş Görüntüler .	23
Şekil 7.10 Deşifrelenmiş Orijinal ve Salt and Pepper Eklenmiş Görüntülerin Benzerlik Grafiği . . . . .	23

Şekil 7.11 Salt and Pepper Ardından Median Filter Uygulanmış Deşifrelenen Görüntü ve Orijinal Deşifrelenmiş Görüntü ile Benzerlik Grafiği .	24
Şekil 7.12 Cropping Saldırısı Farklı Düzeylerde Gerçekleştirilen 3 Görüntü .	25
Şekil 7.13 Cropping Saldırısı Ardından Çözümlenen 3 Görüntü . . . . .	25
Şekil 7.14 Cropping Saldırısı Ardından Çözümlenen 3 Görüntünün Orijinal Çözümlenmiş Görüntü ile Benzerlik Grafikleri . . . . .	26
Şekil 8.1 64x64 Görüntü için Hesaplanan Program Süresi . . . . .	27
Şekil 8.2 Saldırı Sonucu Oluşan Matrix ve Benzerlik Değeri . . . . .	28
Şekil 8.3 Deşifrelenmiş ve Saldırı Sonucu Oluşan Görüntü . . . . .	28
Şekil 8.4 Program Terminal Çıktısı ve Saldırı Sonucu Oluşan Görüntü . . .	29
Şekil 8.5 Salt and Pepper Saldırısı Gerçeklenme Süresi . . . . .	30

## **TABLO LİSTESİ**

---

Tablo 8.1 Farklı Çözünürlükteki Görüntülerin Şifrelenme Süreleri . . . . . 27

## ÖZET

---

# RUBİK KÜPÜ PRENSİBİ İLE ŞİFRELENEN GÖRÜNTÜLERİN KRIPTOANALİZİ

Ömer Buğrahan ÇALIŞKAN

Ömer Aras KAPLAN

Bilgisayar Mühendisliği Bölümü

Bilgisayar Projesi

Danışman: Doç. Dr. Sırma YAVUZ

Kolaylıkla ulaşabileceğimiz birçok görüntü şifreleme algoritması bulunduğundan dolayı, aralarından güvenli bir algoritmayı bulmak zor olabilir. Rubik Küpü Prensibine dayanan bir görüntü şifreleme algoritmasını makalemizin ana odağı olarak aldık. Bu makale, Kritoloji ile ilgili giriş seviyesinde bilgiler ve algoritmanın analizi için kullanılan araçların tanıtımı sağlayacaktır. Ardından makalede görüntü işleme algoritmasının çalışma aşamaları ortaya konup, bahsedilen araçlar kullanılarak algoritmanın ürettiği şifrelenmiş görüntüler test edilecektir. Makale sona erken, bulgular belirtilir ve sonuç kısmında bulgular analiz edilip yorumlanır.

**Anahtar Kelimeler:** Rubik Küpü Prensibi, Kriptoanaliz, Salt and Pepper, Kaba Kuvvet Saldırısı, Median Filtering, Cropping Attack Saldırısı

## **ABSTRACT**

---

# **CRYPTO ANALYSIS OF IMAGES ENCRYPTED WITH THE RUBIC CUBE PRINCIPLE**

Ömer Buğrahan ÇALIŞKAN

Ömer Aras KAPLAN

Department of Computer Engineering

Computer Project

Advisor: Assist. Prof. Sırma YAVUZ

Since there are many image encryption algorithms available, finding a secure one might prove challenging. A certain kind of secure image encryption algorithm which focuses on Rubik's Cube Principle is the main material our work is composed of. This paper provides introduction level information on cryptography and presents a briefing on the tools used in testing. After that, the paper puts an image encryption algorithm under the spotlights by explaining the flow of the algorithm and showing the results of different kinds of attacks. Then at last the paper discloses a conclusion about the state of the algorithm. The aforementioned image encryption algorithm is called A Secure Image Encryption Algorithm Based on Rubik's Cube Principle[1].

**Keywords:** Rubik's Cube Principle, Cryptanalysis, Salt and Pepper Noise, Brute Force Attack, Median Filtering, Cropping Attack

# 1 Giriş

---

Bu bölümde, Şifreleme algoritması tanıtılacak ve hakkında genel kültür bilgileri verilecektir.

## 1.1 Şifreleme Nedir?

Şifreleme veya kriptografi, okunabilir durumdaki bir verinin içeriği bilginin istenmeyen taraflarca anlaşlamayacak bir hale dönüştürülmesinde kullanılan yöntemlerin tümüdür. Şifreleme bir matematiksel yöntemler bütünüdür ve önemli bilgilerin güvenliği için gerekli gizlilik ve kimlik denetimi şartlarını, bilginin içeriğini kaybetmeden aktarılmasını amaçlar. Bu yöntemler, bir bilginin iletimi esnasında ve saklanma süresinde karşılaşabilecek saldırılarından bilgiyi ve bilgiyi paylaşan kişileri koruma amacıyla güdürlüler. Şifreleme, aktarımıabilen her türlü veri çeşidi üzerinde gerçekleştirilebilir.

## 1.2 Görüntü Şifreleme

Görüntü şifreleme ilk olarak 1994 yılında önerilmiş ve basit haliyle görüntü parçalarını uygun anahtar değerleriyle şifrelenmesi esasına dayanmaktadır. 1994'ten bu yana gelişen görüntü şifreleme algoritmaları, şifrelenecek görüntünün çok daha güvenli bir şekilde gizlenmesini ve insan gözüyle ayırt edilemeyecek duruma gelmesine imkan sağlamıştır.

Bir görüntü şifreleme algoritmasının tercih edilebilmesi için şifrelenmiş görüntünün orijinal görüntüyle olan korelasyon değerinin sıfıra yakın olması, kriptolojide kullanılan saldırı tiplerine karşı dayanıklı olması, kaba-kuvvet saldırılara uzun süre dayanabilmesi ve çözümlenmiş görüntünün orijinal görüntüsünden farkının olabildiğince az olması gerekmektedir.

### **1.3 Rubik Küpü Görüntü Şifreleme Algoritması**

Bu projede gerçekleştirilecek olan Rubik Küpü Görüntü Şifreleme Algoritması:

En başta orijinal grayscale görüntü Rubik Küpü prensibi kullanılarak karıştırılır. Bu aşamada, piksellerin sadece yerini değiştirir, piksel değerleri değiştirilmez. Sonrasında, iki rastgele gizli anahtar dizisi kullanılarak satır ve sütunlara bitwise-XOR( $\oplus$ ) işlemi uygulanır. Ardından, tekrar satır ve sütunlara anahtar dizisinin ters çevrilmiş hali ile bitwise-XOR( $\oplus$ ) işlemi uygulanır. Son olarak, bu adımlar istenilen güvenlik seviyesine göre tekrar edilerek güvenliği arttırılabilir.[1]

Çözümleme aşamasında, şifreleme aşamasında üretilen rastgele dizilerin kullanılması gereklidir. Çözümleme adımları, şifreleme aşamasının adımlarının tersten takip edilmesiyle gerçekleştirilebilir ve belirlenen iterasyon sayısına bağlı olarak tekrar edilir.[1]

## 2 Ön İnceleme

---

Ön inceleme bölümü, projenin yapılacak alanda daha önceden yapılmış olan mevcut ve benzeri çalışmaların incelenmesini içerir.

### 2.1 Rübik'in Kübü Prensibine Dayanan Görüntü Şifreleme Algoritması

Projemizin referans aldığı ve 3 araştırmacı tarafından ortaya konan "A Secure Image Encryption Algorithm Based on Rubik's Cube Principle"[1] makalesini inceledik. Bu şifreleme algoritması projemizin referans aldığı ana kaynaktır. Görüntü rastgele oluşturulan diziler aracılığıyla kaydırma işlemi yapılarak şifrelenmektedir. Diziler rastgele olduğundan görüntünün deşifre edilebilmesi için bu dizilere sahip olunması gerekmektedir. Makalede algoritma üzerinde test edilmiş, salt & pepper noise, speckle-noise ve cropping attack saldırıları yer almaktadır.

### 2.2 Geliştirilmiş Hiperkaotik Dizilere Dayalı Görüntü Şifreleme Algoritması

Bir diğer makalede "Cryptanalysis of a novel image encryption scheme based on improved hyperchaotic sequences"[2] seçilen görüntü şifreleme algoritması üzerinde kriptoanaliz yapılmıştır. Şifreleme algoritması incelemiştiğinde başlangıç aşamasında 4 adet sabit değer üretilir ve bu değerler birbirleriyle matematiksel işleme sokularak yeni değerler elde edilir. Bu yeni değerler ile bir kaotik anahtar dizisi oluşturulur ve görüntü matrisi ile XOR ve mod işlemlerine sokulur. İşlemler sonucunda şifrelenmiş görüntü elde edilmiş olur. Makalede, chosen-plaintext saldırısı kullanılmış ve gizli anahtarlar başarıyla ele geçirilmiştir.

## **2.3 Dairesel Pikseller Arası Bit Seviyesi Permütasyonu ile Kaos Tabanlı Görüntü Şifreleme Düzeninin Kriptanaliz ve İyileştirilmesi**

Bir diğer makalede "Cryptoanalysis and Improvement of Chaos-Based Image Encryption Scheme with Circular Inter-Intra-Pixels Bit-Level Permutation"[3] seçilen görüntü şifreleme algoritması üzerinde kriptoanaliz yapılmıştır. Algoritmanın orijinal hali 2 aşamadan oluşur. İlk aşama olan permütasyon aşamasında pikseller bit düzeyinde dairesel kaydırma ile karıştırılır ve yeni piksel değerleri oluşturulur, difüzyon aşamasında ise permütasyon işlemine uğrayan matris iki şifrelenmiş matris kullanılarak şifrelenir. Bu geliştirilmiş algoritmada ise orijinal algoritmada kullanılan dairesel kaydırma işlemi değiştirilerek kaydırma adımları ve yönleri rastgele gerçekleştirilir. Makalede güvenlik açığı analiz edilirken chosen-plaintext saldırısı kullanılmış ve geliştirilmiş yöntem sayesinde bu saldırının hala algoritmayı kıramasına rağmen güvenilirliğini önemli ölçüde arttırmıştır. Makale içerisinde işlenen konular ve analiz yöntemleri projemiz ile paralellik gösterdiğiinden bu makale projemiz için yol gösterici bir örnek olacaktır.

## **2.4 Kaldırma Saldırılarının Sayısal Görüntü Filigranları Üzerindeki Etkisinin Uzamsal ve Frekans Alanı Analizi**

Bir diğer makalede "A Spatial and Frequency Domain Analysis of the Effect of Removal Attacks on Digital Image Watermarks"[4] görüntü üzerine uygulanan çeşitli saldırıların etkileri ve zarara uğrayan görüntülerin zarar etkilerinin giderilebilmesi için yapılabilecek yöntemlerden bahsedilmiştir. Projemizde uyguladığımız Salt and Pepper saldırısının, saldırının öncesi hazırlık aşamasında nasıl uygulandığı, nasıl etkiler gösterdiği, etkilerin nasıl azaltılabileceğine dair bilgiler bu makaleden elde edilmiştir. Makalede Salt and Pepper saldırısı uygulanmış bir görüntüden bu etkiyi azaltabilmek için uygulanan Median Filtering yönteminden bahsedilmiş ve analizi yapılmıştır. Bizim projemizde Salt and Pepper saldırısı ve median filtering yöntemi uygulanmış ve analizi yapılmıştır. Bu makalede verilen bilgiler ve analizler proje çalışmamız için önemli bir kaynak olmuştur.

## **2.5 Görüntü Şifreleme Teknikleri Üzerine Bir Literatür İncelemesi**

Bir diğer makalede "A Literature Review on Image Encryption Techniques" [5] görüntü şifreleme işlemlerinin temel olarak nasıl gerçekleştiğinden, farklı şifreleme tekniklerinden ve bu tekniklerin analizleri sonucu karşılaştırılmasından bahsedilmiştir. Görüntü şifrelemenin prensiplerine, farklı şifreleme yöntemlerine ve bu yöntemlerin olumlu veya olumsuz sonuçlarını ele alabilmek amacıyla bu makale incelenmiştir. Birden çok yöntem gözden geçirilmiş ve analiz sonuçları göz önüne alınmıştır. Projeye öncesinde ve devamında görüntü şifreleme konusu adına bilgiler edinmek amacıyla bu makale referans aldığımız kaynaklarımıza arasında bulunmaktadır.

## **2.6 L1-L2 TV Modelini Kullanarak Medyan Filtreleme için Geri Döndürme Tekniği**

Bir diğer makalede "Anti-Forensic Technique for Median Filtering using L1-L2 TV Model" [6] Median Filtering yönteminden bahsedilmiş ve teorik olarak açıklanmıştır. Makalede bulanıklaştırılan bir görüntünün eski haline nasıl döndürülebileceği konusunda çalışmalar yapılmış ve elde edilen veriler analiz edilmiştir. Biz projemizde Median Filtering yöntemini kullanacağımızdan dolayı bu makale bu yöntemi anlayabilmek ve implemente edebilmek adına yararlı bir kaynaktır.

# 3

## Fizibilite

---

### 3.1 Teknik Fizibilite

Seçilen bir görüntü dosyasının Rübik Küpü algoritmasına göre yazılmış kod içerisinde şifrelenmesi ve ardından şifrelenmiş görüntünün çözümlenmesi istenmektedir.

Proje PyCharm üzerinden Python programlama dili versiyon 3.9 ile kodlanacaktır.

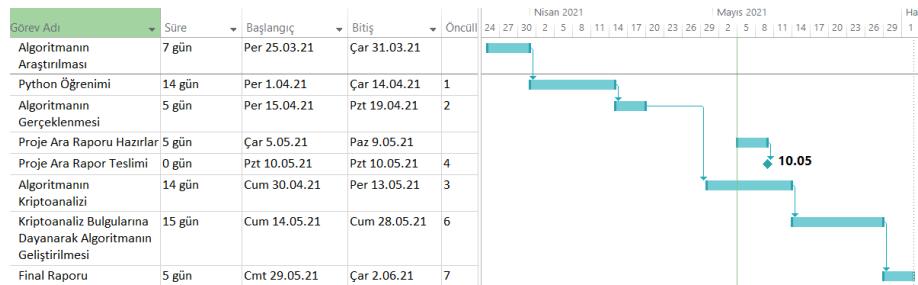
```
import numpy as np
import random
import PIL.Image
import PIL
import matplotlib.pyplot as plt
```

Şekil 3.1 Python Kütüphaneleri

Projede Python kütüphanelerinin sağlayacağı birden çok avantajdan faydalanyılmıştır. Görüntünün okunabilmesi için Pillow, Görüntünün matrise aktarılıp matematiksel işlemlerin yapılabilmesi için NumPy, Orijinal görüntünün, şifrelenmiş görüntünün ve çözümlenmiş görüntünün benzerlik katsayılarının grafiksel olarak görüntülenebilmesi için matplotlib, Üretilerek rastgele sayı dizileri için random kütüphanesi kullanılmıştır. Cripto-analiz aşamasında görüntü dosyasının okunabilmesi için cv2 kütüphanesi, programın çalışma süresinin hesaplanabilmesi için time kütüphanesi kullanılmıştır.

Proje Intel Core i7-7700HQ 2.80 GHz İşlemci, 16 GB Ram ve Windows 10 İşletim Sisteminde gerçekleşmiştir.

## 3.2 İş Gücü ve Zaman Planlanması



Şekil 3.2 Gantt Diyagramı

Proje gerçekleştirirken şelale yazılım geliştirme modeli kullanılmıştır.

## 3.3 Ekonomik Fizibilite

Projede kullanılan kütüphane ve IDE tamamen ücretsiz olup herhangi bir maliyeti bulunmamaktadır.

Projenin 2 kişi tarafından 70 iş gününde tamamlanması planlanmıştır. Her bir geliştirici için 30 günlük ödenmesi gereken tutarın 4500 TL olması gerektiği belirlenmiştir. Yapılacak hesaplama ile 2 geliştiricinin 70 günlük maliyeti 21.000 TL olacaktır.

# 4

## Sistem Analizi

---

Bu bölümde hedefler ve sistem çalışması açıklanacaktır.

### 4.1 Hedefler

Projemizde gerçekleştirilecek birden fazla hedef bulunmaktadır, bunlar:

- Rubik Küpü Prensibi Baz Alınan Görüntü Şifreleme Algoritmasının Şifreleme ve Deşifreleme Aşamalarının Gerçeklenmesi.
- Şifreleme Algoritmasının Zayıflıklarının Çeşitli Cripto Saldırıları ile Analizi.
- Analiz sonucu elde edilen verilerin gözden geçirilmesi ve algoritmanın güvenilirliğinin değerlendirilmesi.

Geçeklenecek şifreleme ve deşifreleme aşamaları python programlama diliyle kodlanacak ve gerekli kütüphanelerin kullanılmasıyla desteklenecektir. Kodlama aşaması proje üyeleri tarafından özgün olarak yazılacaktır. Program çalıştırıldığında şifrelenen görüntü, deşifrelenen görüntü ve bu görüntülerin birbirleriyle olan benzerlik katsayısı grafiği görüntü olarak kaydedilecek ve kullanıcıya gösterilecektir. Programın terminal çıktısında orijinal, şifrelenen ve deşifrelenen görüntünün matris değerleri, benzerlik katsayısı, rastgele oluşturulan sayı dizisi ve aşamaları çıktı olarak gösterilmektedir.

Kripto Analiz aşamasında ise kaba-kuvvet, salt and pepper ve cropping attack saldırıları gerçekleştirilecek ve algoritmanın dayanıklılığı test edilecektir. Bu saldırıların gerçekleştirilebilmesi için gerekli kodlar bizim tarafımızdan python ortamında yazılacaktır ve çıktıları analiz edilerek gösterilecektir.

## 4.2 Sistem Çalışması

İlk olarak programda görüntü dosyasının dizini girilir ve bu görüntü program içerisinde grayscale haline dönüştürülür. Ardından görüntünün değerleri matrise aktarılır.

```
def read_image(path):
    try:
        image = PIL.Image.open(path)
        return image
    except Exception as e:
        print(e)

def convert_to_grayscale(image):
    grayscale = image.convert("L")
    return grayscale

path = "lena.png"
I_0 = read_image(path)

mode_to_bpp = {'1': 1, 'L': 8, 'P': 8, 'RGB': 24, 'RGBA': 32, 'CMYK': 32, 'YCbCr': 24, 'I': 32, 'F': 32}
grayscale = convert_to_grayscale(I_0)

I_0 = grayscale
img_array = np.array(grayscale)
org_array = np.copy(img_array)
bpp = mode_to_bpp[I_0.mode]
# Bitsize bulundu. Grayscale için genellikle bpp=8
```

Şekil 4.1 Image Dosyasının Okunması

Şifreleme algoritmasının çalışabilmesi için iki adet rastgele sayılardan oluşan dizi oluşturulur(KR ve KC).

```
# DEFINE KC AND KR
for i in range(0, M):
    KR.insert(i, random.randint(0, (2 ** bpp) - 1))
for j in range(0, N): # prep
    KC.insert(j, random.randint(0, (2 ** bpp) - 1))
```

Şekil 4.2 KR ve KC Dizilerinin Oluşturulması

Bu dizi değerlerine bağlı olarak görüntü matrisinin değerleri satır ve sütun bazında kaydırılır.

```
# SECTION 4-5 SUM AND SHIFT
for i in range(0, M): # 4
    Ma = matrix_sum(img_array, i, 0) % 2 # 4a 4b
    img_array[i] = row_shift_by(Ma, abs(KR[i] % M), img_array, i) # 4c

for i in range(0, N): # 5
    Mb = matrix_sum(img_array, i, 1) % 2 # 5a 5b7
    img_array = col_shift_by(Mb, abs(KC[i] % N), img_array, i)
```

Şekil 4.3 Kaydırma İşlemleri

```

def col_shift_by(direction, num, array, colno):
    length = len(array)
    temp = np.arange(length)
    if direction == 0: # shift down
        for j in range(0, length):
            temp[j] = array[(j - num + length) % length][colno]
    else: # shift up
        for j in range(0, length):
            temp[j] = array[(j + num) % length][colno]
    for j in range(0, length):
        array[j][colno] = temp[j]
    return array

def row_shift_by(direction, num, array, rowno):
    length = len(array[0])
    temp = np.arange(length)
    if direction == 0: # shift right
        for j in range(0, length):
            temp[j] = array[rowno][(j - num + length) % length]
    else: # shift left
        for j in range(0, length):
            temp[j] = array[rowno][(j + num) % length]
    return temp

```

**Şekil 4.4** Kaydırma Fonksiyonları

Ardından matris değerleri bulunduğu konum kontrol edilerek KR ve KC dizileriyle veya bu dizilerin ters çevrilmiş haliyle (revKR, revKC) XOR işlemine sokulur. Bu işlemler belirlenen iterasyon sayısı ile tekrar edilir.

```

def xor_operation(array, KRC, revKRC, i, j):
    if (i % 2) == 0:
        return array[i][j] ^ revKRC[j]
    else:
        return array[i][j] ^ KRC[j]

```

**Şekil 4.5** XOR Fonksiyonu

```

# SECTION 6 XOR OPERATIONS
revKC = KC[::-1]
revKR = KR[::-1]

for i in range(0, M):
    for j in range(0, N):
        img_array[i][j] = xor_operation(img_array, KR, revKR, i, j)

for j in range(0, N):
    for i in range(0, M):
        img_array[i][j] = xor_operation(img_array, KC, revKC, i, j)

```

**Şekil 4.6** XOR İşlemleri

Bu adımda şifrelenmiş görüntü elde edilmiş olur.

Çözümleme adımda ise şifreleme adımda yapılan işlemler tersten izlenir. İlk adımda, şifrelemenin son adımda yapılan XOR işlemi yapılır ve anlaşıldığı üzere şifrelemede yapılan işlem ortadan kaldırılır. Ardından şifrelemenin ilk adımda yapılan satır ve sütunların kaydırılması işlemi tersten tekrarlanır, bu adımda yapılmasıyla aslında şifreleme adımda gerçekleştirilen bütün adımlar ortadan kaldırılmış olur ve orijinal görüntü matrisi elde edilir. Çözümlemenin yapılabilmesi için şifreleme adımda oluşturulan KR, KC dizilerine ve toplam iterasyon sayısına ihtiyaç duyulmaktadır.

```
# DECRYPT XOR OPERATIONS

for j in range(0, N):
    for i in range(0, M):
        img_array[i][j] = xor_operation(img_array, KR, revKR, i, j)

for i in range(0, N):
    for j in range(0, N):
        img_array[i][j] = xor_operation(img_array, KC, revKC, i, j)

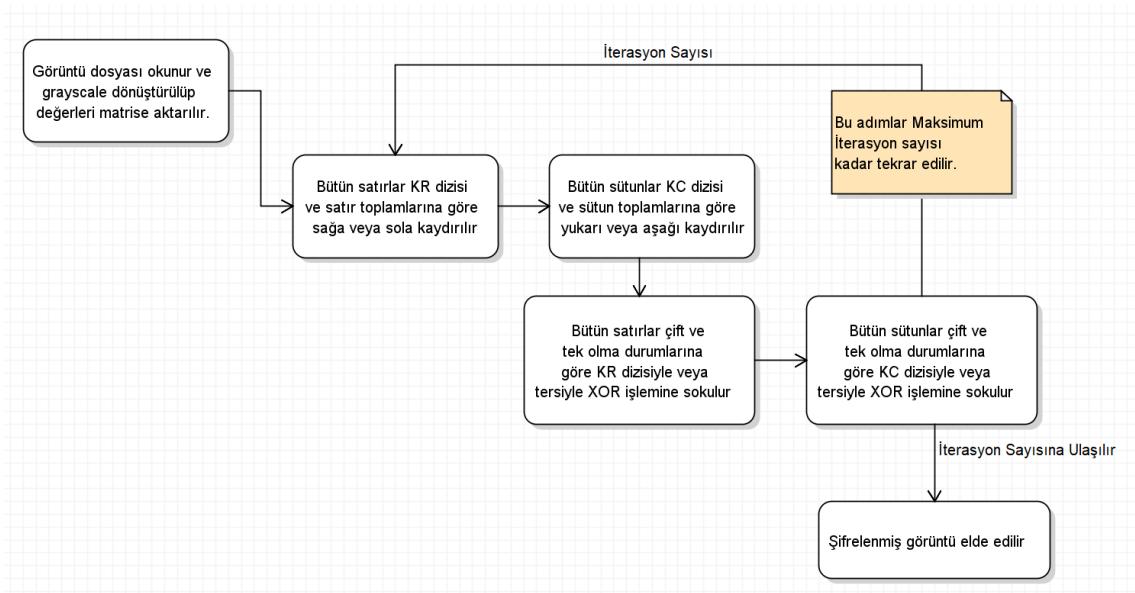
# DECRYPT SUM-SHIFT OPERATIONS !!

for i in range(0, N):
    Mb = matrix_sum(img_array, i, 1) % 2
    img_array = col_shift_by(not Mb, abs(KC[i] % N), img_array, i)

for i in range(0, M):
    Ma = matrix_sum(img_array, i, 0) % 2
    img_array[i] = row_shift_by(not Ma, abs(KR[i] % M), img_array, i)
```

**Şekil 4.7** Deşifreleme İşlemleri

Yazılacak olan program buraya kadar bahsedilen aşamaları başarıyla gerçekleştirmektedir. Buradan sonraki adımlarda ise bu algoritmanın kripto-analizi yapılacak ve güvenilirliği test edilecektir. Son aşamada ise yapılan testlerin ardından elde edilecek sonuçlara dayanarak algoritmanın geliştirilebilmesi için önerilerde bulunulacaktır.



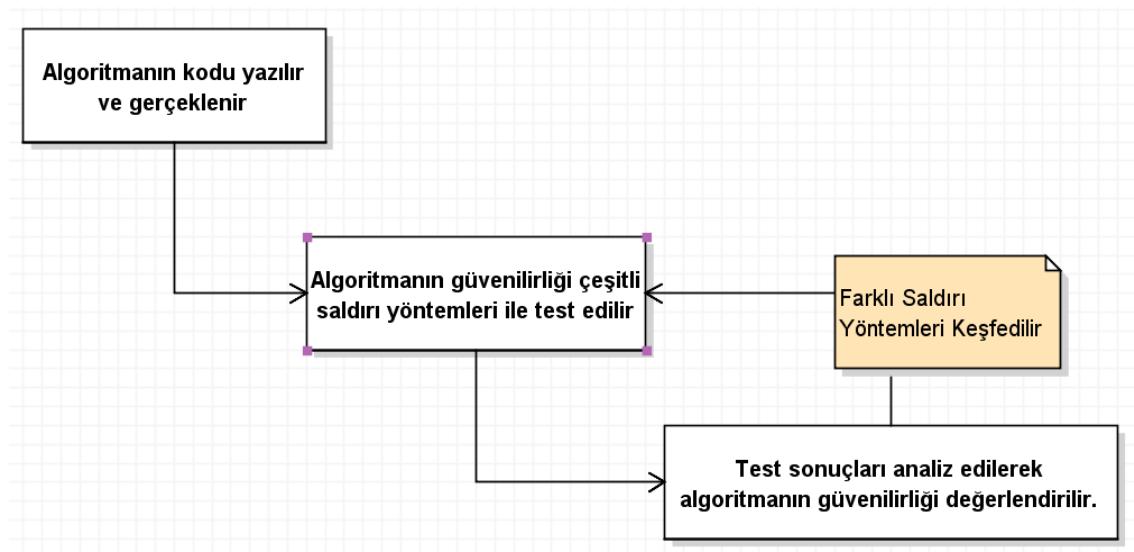
**Şekil 4.8** Algoritma Akış Şeması

# 5 Sistem Tasarımı

Bu bölümde sistemin yazılım tasarımı incelenecaktır.

## 5.1 Yazılım Süreci

Programımız 3 ana adımdan oluşmaktadır ve bu adımlar birbirini takip etmektedir.



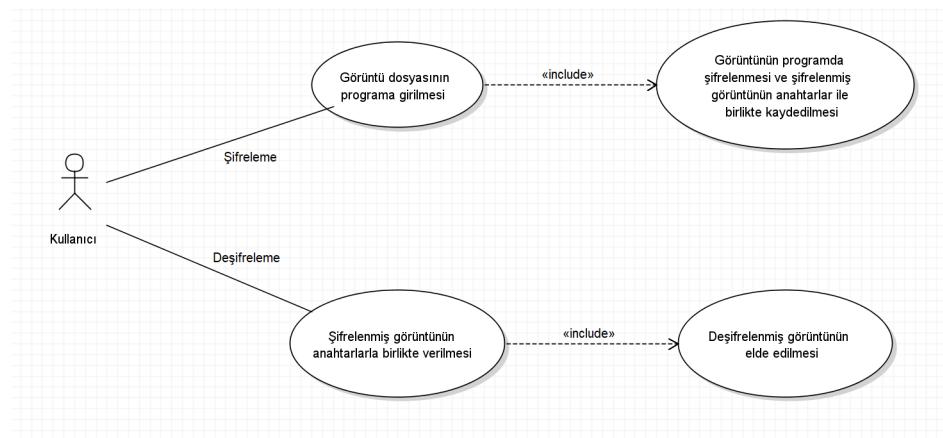
Şekil 5.1 Yazılım Akış Şeması

Referans aldığımız algoritmanın yapılan testler sonucu elde edilen bulgularına dayanarak güvenilirliğinin test edilmesi amaçlanmaktadır.

## 5.2 Program Kullanımı

Program 2 taraflı çalışmaktadır, bunlar şifreleme ve deşifreleme adımlarıdır. Uygulama çalıştırıldığında şifreleme seçilir ise kullanıcidan şifreleyeceği görüntünün dosya dizini istenir ardından belirtilen dizine şifrelenmiş dosya ile beraber KR ve KC dizileri de kaydedilir. Deşifrelemede ise kullanıcidan şifrelenmiş görüntü dosyası ile

beraber KR ve KC dizilerinin bilgisinin olduğu bir dosya girmesi istenir.



**Şekil 5.2** Program Kullanıcı Şeması

# 6

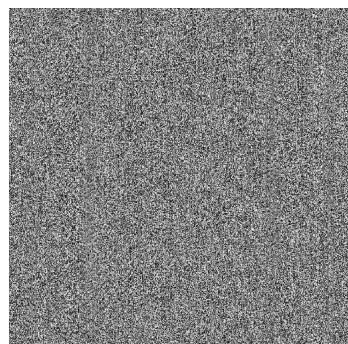
## Uygulama

---

Program başlatıldığında dosya dizini girilen görüntü açılır ve işlenir. Programda örnek olarak "lena" görüntüsü kullanıldı. Programın çıktıları ve elde edilen görüntüler aşağıda listelenmiştir.



**Şekil 6.1** Lena Orijinal Görüntü



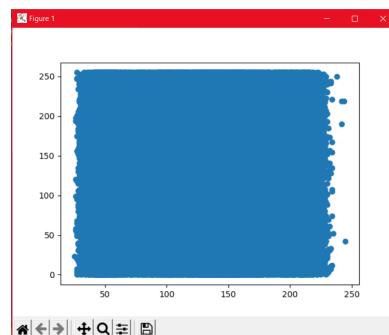
**Şekil 6.2** Şifrelenmiş Lena Görüntüsü



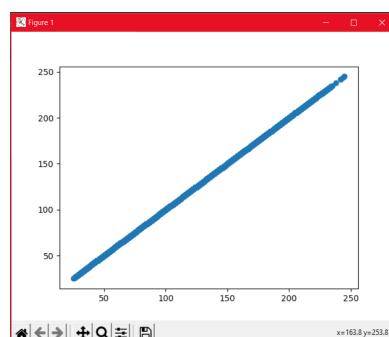
Şekil 6.3 Deşifrelenmiş Lena Görüntüsü

Görüntü algoritma adımlarına uygun olarak şifrelenir. Şifrelenen görüntü kaydedilir. Şifrelenen görüntü matrisi program terminal çıkışında gösterilir. Şifrelenmiş görüntü KR, KC ve iterasyon sayısı değerlerini kullanarak şifrelenmiş görüntüyü orijinal haline getirir fakat ilk adımda orijinal görüntüyü grayscale dönüştürdüğümüz için deşifrelenmiş görüntü de grayscale olacaktır.

Görüntüler arası benzerlikler.



Şekil 6.4 Orijinal Lena - Şifrelenmiş Lena



Şekil 6.5 Orijinal Lena - Deşifrelenmiş Lena

Görüntü matrisleri arasında benzerlik hesaplamaları yapılmış ve şekillerdeki grafikler elde edilmiştir. Grafiklerin incelendiğinde orijinal ve şifrelenmiş görüntüler arasında benzerliğin olmadığı neredeyse tamamen ayrik olduğu, orijinal ve deşifrelenmiş görüntülerin ise neredeyse kayıpsız bir şekilde eşleştiği görülmektedir.

Son olarak programın terminal çıktısı şekilde gösterildiği gibidir.

```
---- Original Matrix:
[[162 162 162 ... 170 155 128]
 [162 162 162 ... 170 155 128]
 [162 162 162 ... 170 155 128]
 ...
 [ 43  43  50 ... 104 100  98]
 [ 44  44  55 ... 104 105 108]
 [ 44  44  55 ... 104 105 108]]
---- Encrypted Matrix:
[[120  77 228 ... 137 119  38]
 [ 43  90 114 ...  72  17 137]
 [163 139 244 ...  99 165 223]
 ...
 [ 44  45  59 ... 253  77 142]
 [114 176  67 ... 187 111  41]
 [197 132 204 ... 236 135 179]]
KR: [176, 101, 68, 23, 168, 205, 226, 214, 105, 129, 172, 205, 220, 91, 10, 113
KC: [237, 151, 76, 80, 38, 128, 141, 145, 54, 192, 118, 176, 40, 90, 112, 196,
----- Decryption -----
---- Decrypted Matrix:
[[162 162 162 ... 170 155 128]
 [162 162 162 ... 170 155 128]
 [162 162 162 ... 170 155 128]
 ...
 [ 43  43  50 ... 104 100  98]
 [ 44  44  55 ... 104 105 108]
 [ 44  44  55 ... 104 105 108]]
Correlation Between Encrypted and Original Matrix: -0.004116684859084035
Correlation Between Decrypted and Original Matrix: 1.0
```

Şekil 6.6 Program Terminal Çıktısı

# 7

## Deneysel Sonuçlar

Bu bölümde şifrelenen görüntüye uygulanan saldırılar analiz edilecektir.

### 7.1 Kaba-Kuvvet Saldırı

Şifreleme işlemi gerçekleştirildikten sonra oluşan "enc\_lena.png" dosyası şifrelenmiş görüntüyü temsil etmektedir. Algoritmanın deşifreleme aşamasında bahsedildiği üzere şifrelenen görüntünün çözümlenebilmesi için rastgele oluşturulan KR ve KC dizilerinin olması gerekmektedir.

Saldırı senaryomuzda bu dizilerin olmadığı varsayılarak rastgele dizi üretimi yapılmış ve deşifreleme sonucu oluşması gereken görüntü ile benzerlik değeri karşılaştırılmış ve belirli bir değere ulaşınca saldırının sonlandırılması planlanmıştır.

```
path_enc = "enc_lena.png"
path_dec = "dec_lena.png"

I0_enc = read_image(path_enc)
I0_dec = read_image(path_dec)

img_array = np.array(I0_enc)
org_array = np.array(I0_dec)
bpp = 8
```

**Şekil 7.1** Kaba-Kuvvet Saldırı Görüntü Tanımlamaları

Görüntünün açılması ve matrise çevrilmesinin ardından şifreleme aşamasında üretilmesi ve deşifreleme aşaması için gerekli olan KR ve KC dizileri, kaba-kuvvet saldırı senaryosunun gerçekleştirilebilmesi için rastgele üretilmiştir.

```

while correlation < 0.5:
    # DEFINE KC AND KR
    for i in range(0, M):
        KR.insert(i, random.randint(0, 255))
    for j in range(0, N): # prep
        KC.insert(j, random.randint(0, 255))

```

**Şekil 7.2** Rastgele Üretilen KR ve KC dizileri

Programın döngüye girebilmesi ve belirlenen benzerlik değerine ulaşana kadar devam edebilmesi için "correlation" değeri başlangıçta 0 olarak girilmiş ve döngü süresince üretilen diziler ile yapılan işlemler sonucunda güncellenmiştir.

```

revKC = KC[::-1]
revKR = KR[::-1]

for j in range(0, N):
    for i in range(0, M):
        img_array[i][j] = xor_operation(img_array, KR, revKR, i, j)

for i in range(0, M):
    for j in range(0, N):
        img_array[i][j] = xor_operation(img_array, KC, revKC, i, j)

# DECRYPT SUM-SHIFT OPERATIONS !!

for i in range(0, N):
    Mb = matrix_sum(img_array, i, 1) % 2
    img_array = col_shift_by(not Mb, abs(KC[i] % N), img_array, i)

for i in range(0, M):
    Ma = matrix_sum(img_array, i, 0) % 2
    img_array[i] = row_shift_by(not Ma, abs(KR[i] % M), img_array, i)

print("---- Cracked Matrix: \n" + format(img_array))

```

**Şekil 7.3** Kaba-Kuvvet Saldırı İşlemleri

Kaba-kuvvet saldırılarında kullanılan fonksiyonlar ve algoritmanın temel yapısı deşifreleme kodlarıyla aynıdır. Algoritma rastgele üretilen değerlerin, şifrelenmiş görüntü üzerinde yapılan XOR ve kaydırma işlemleri sonucunda deşifrelenmiş görüntüyle benzerlik göstermesi halinde sonlandırılması temeline dayandırılmıştır.

```

plt.scatter(img_array, org_array)
corr2 = np.corrcoef(org_array.flat, img_array.flat)
correlation = corr2[0, 1]
k += 1
print(format(corr2[0, 1]))

```

**Şekil 7.4** Kaba-Kuvvet Saldırı Döngü Değişkeni

Döngünün son adımda hesaplanan benzerlik değeri, döngü tanımlanırken belirlenen benzerlik değerinden küçük ise döngü devam eder. Her adımin görüntülenebilmesi açısından oluşturulan "k" değeri adım sayısını temsil eder, programın terminal çıktısında ise rastgele oluşturulan dizi değerlerine göre yapılan deşifreleme işlemi sonucunda oluşan görüntü matrisi gösterilir.

```

plt.show()
savepath = "attack_lena.png"
dec_photo = PIL.Image.fromarray(img_array)
dec_photo = dec_photo.save(savepath)

print("\nCorrelation Between Decrypted and Original Matrix: " + format(corr2[0, 1]))

```

**Şekil 7.5** Kaba-Kuvvet Saldırı Başarı Durumu

Benzerlik değerinin belirlenen değerin üzerine çıkabilmesi durumunda döngü sonlandırılır, iki görüntünün benzerlik grafiği gösterilir ve kırılmış görüntü resim dosyası halinde kaydedilir. Son olarak benzerlik değerinin terminal çıktısına yazdırılmasıyla program sonlandırılır.

Kaba-kuvvet saldırımız çeşitli şifrelenmiş görüntüler üzerinde analiz edildiğinde görüntünün boyutları büyüğünde rastgele oluşturulan dizilerin boyutu artacağından görüntünün kırılma ihtimalinin de oldukça azaldığı görülmüştür. Saldırı senaryolarımızda kullanılan referans görüntü "lena" görüntüsüdür. Lena görüntüsünün orijinal boyutu 512x512 olmakla birlikte analiz sürecinde 64x64, 32x32 ve 16x16 boyutundaki görüntüler üzerinde saldırı gerçekleştirılmıştır.

Saldırının alışlagelmiş kaba-kuvvet saldırılardan önemli bir farkı bulunmaktadır. Şifrelenmiş görüntüyü kırmak için oluşturulacak olan KR ve KC dizilerinin bütün olasılıkları gezebilmesi için 16x16 çözünürlükte bir görüntü ele alındığında  $256^{16}$  farklı değer KR dizisi için,  $256^{16}$  farklı değer de KC dizisi için hesaplanacaktır. Veriler göz önüne alındığında toplam kontrol edilmesi gereken  $1.1579209e+77$  ( $256^{32}$ ) değer olduğu görülmüş ve bu değerin hesaplanması saniyede 1 milyon(1.000.000) işlem yapabilen bir bilgisayar için bile  $3.66930829 \times 10^{63}$  yıl sürecektir. Bu yüksek değer göz önüne alındığında kaba-kuvvet saldırının aynı şifreleme aşamasında olduğu gibi

rastgele değer üretilerek denenmesi gerektiğine karar verilmiştir.

Bahsedilen 256<sup>32</sup> olasılık için geliştirilmiş bir kaba-kuvvet saldırısı senaryosu kullanılmak istenildiğinde üretilecek değerlerin bir regresyon yöntemiyle olması gereken değerlere yaklaşması öngörmektedir fakat işlenmesi gereken ham verinin çok büyük olması bu senaryonun verimsiz olma sonucunu doğuracaktır.

## 7.2 Salt and Pepper Saldırısı

Salt and Pepper, orijinal görüntü üzerindeki piksellere tamamen siyah veya tamamen beyaz renge sahip bitlerin rastgele dağıtılmaması esasına dayanır.

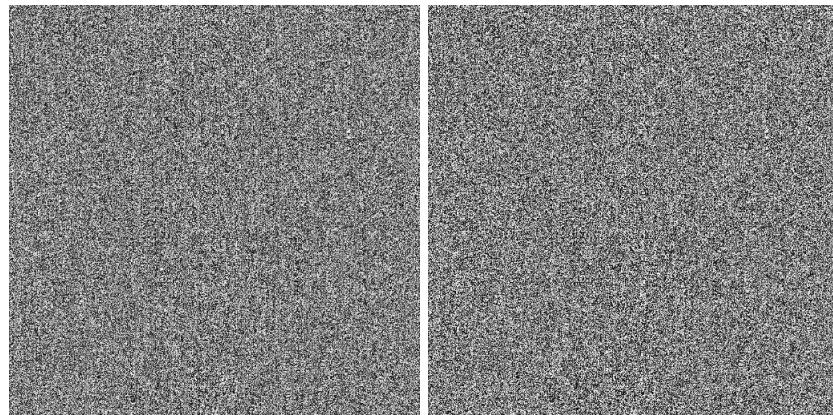
Bu saldırının uygulanma aşaması ise 4 ana adımdan oluşur. Bunlar sırasıyla aşağıdaki şekilde verilmiştir.

- Görüntü şifrelenir.
- Şifrelenmiş görüntüye salt and pepper uygulanır.
- Şifrelenmiş ve salt and pepper eklenmiş görüntü deşifreleme işlemeye sokulur.
- Deşifrelenen görüntü üzerinden salt and pepper etkisi median filtering yöntemiyle azaltılarak elde edilen görüntülerin karşılaştırması yapılır.

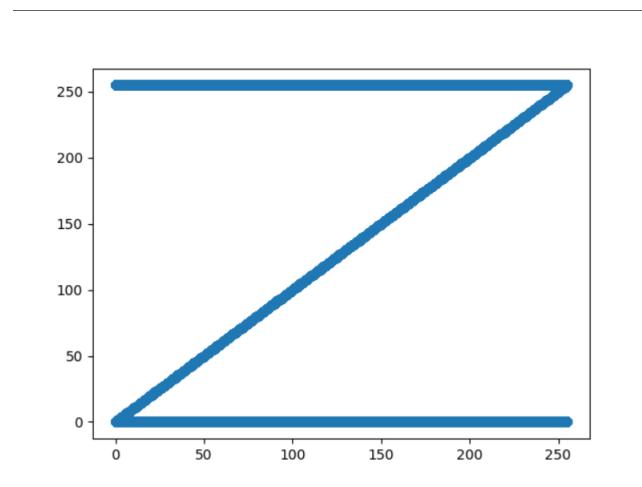
Rubik'in kübü prensibine göre şifrelenen 512x512 çözünürlükteki lena görüntüsü aşağıdaki fonksiyondan geçirilerek üzerine salt and pepper eklenir. Örneğimizde salt and pepper değeri 0.9 olarak belirlenmiş ve uygulanmıştır. 0.9 değeri eklenecek salt and pepper'in miktarını belirlemektedir ve bu sayı 0'a yaklaştıkça eklenen salt and pepper artar yani verilen değer 0.1 olduğunda görüntü daha karmaşık gözükmemektedir..

```
def salt_pepper(img, snr):
    img_ = img.copy()
    c, h, w = img_.shape
    temp = np.random.choice((0, 1, 2), size=(1, h, w), p=[snr, (1 - snr) / 2., (1 - snr) / 2.])
    temp = np.repeat(temp, c, axis=0)
    img_[temp == 1] = 255
    img_[temp == 2] = 0
    return img_
```

Şekil 7.6 Görüntüye Salt and Pepper Ekleme Fonksiyonu



**Şekil 7.7 Şifrelenmiş ve Salt and Pepper Eklenmiş Görüntü**



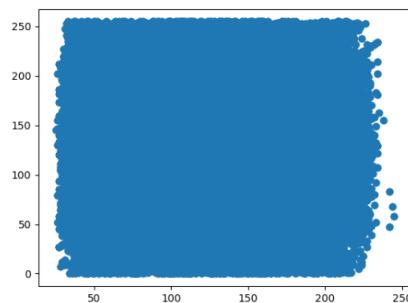
**Şekil 7.8 Şifrelenmiş ve Salt and Pepper Eklenmiş Görüntülerin Benzerlik Grafiği**

Şifrelenmiş ve Salt and Pepper eklenmiş görüntünün benzerlik değeri python ortamında hesaplanmış olup yaklaşık olarak 0.8269 bulunmuştur.

Bir sonraki adımda şifrelenmiş ve salt and pepper eklenmiş görüntünün deşifreleme sonrası görüntüsü ve orijinal şifrelenmiş görüntünün salt and pepper eklenmeden oluşan deşifrelenmiş görüntüsünün karşılaştırılması aşağıda yapılmıştır.



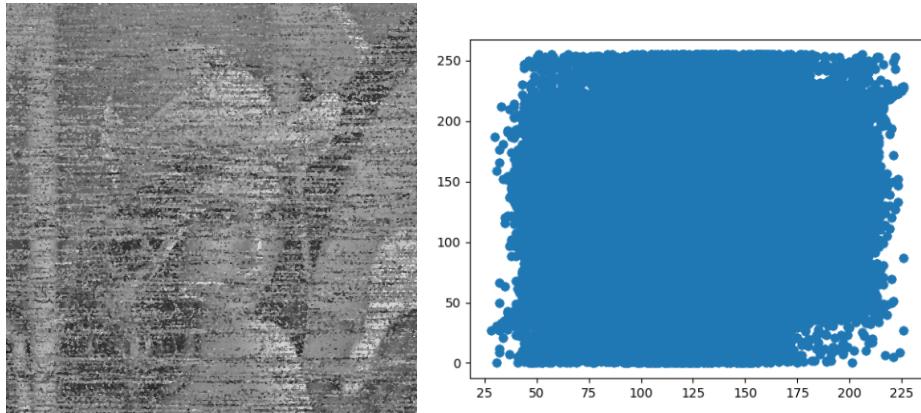
**Şekil 7.9** Deşifrelenmiş Orijinal ve Salt and Pepper Eklenmiş Görüntüler



**Şekil 7.10** Deşifrelenmiş Orijinal ve Salt and Pepper Eklenmiş Görüntülerin Benzerlik Grafiği

Orijinal hali çözümlemiş ve Salt and Pepper eklenen hali çözümlemiş görüntünün benzerlik değeri python ortamında hesaplanmış olup yaklaşık olarak 0.2121 bulunmuştur.

Son aşamada ise Salt and Pepper işlemi yapıldıktan sonra çözümlemiş görüntü üzerine Median Filtering uygulanır. Bu işlem Salt and Pepper saldırısını etkisi azaltır. Bu işlemlerin ardından elde edilen sonuçlar analiz edilmiştir. Median Filtering işleminin uygulanabilmesi için python cv2 kütüphanesinde bulunan medianBlur fonksiyonu kullanılmıştır. Bu fonksiyon görüntü üzerinde oluşmuş Salt and Pepper içeren bitlerin bulanıklaştırılması işlemini gerçekleştirir. Aşağıdaki görüntülerde Salt and Pepper etkisi azaltılmış görüntü ve bu görüntü ile orijinal çözümlemiş görüntünün benzerlik grafiği gösterilmiştir.



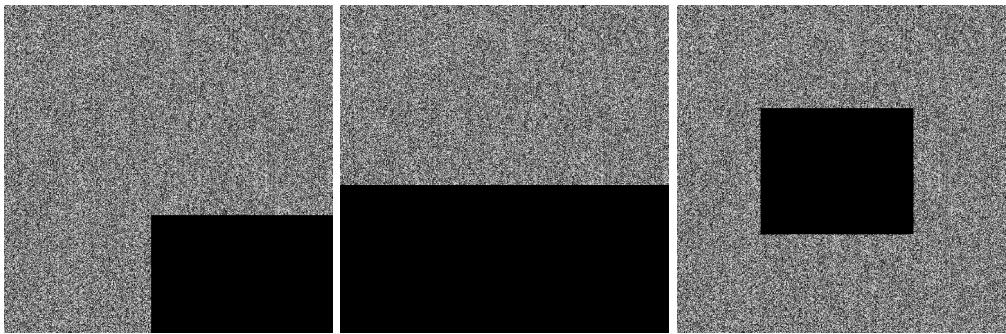
**Şekil 7.11** Salt and Pepper Ardından Median Filter Uygulanmış Deşifrelenen Görüntü ve Orijinal Deşifrelenmiş Görüntü ile Benzerlik Grafiği

Orijinal hali çözümlenmiş ve Salt and Pepper eklenen hali çözümlendikten sonra Salt and Pepper etkisi azaltılan görüntünün benzerlik değeri python ortamında hesaplanmış olup yaklaşık olarak 0.3852 bulunmuştur.

Bu iki değerden görülebileceği üzere Salt and Pepper işlemi görüntü şifreleme algoritmalarına büyük zarar vermektedir. Bu işlem yapılmadığı takdirde benzerlik değeri yaklaşık olarak 1 olacaktır çünkü bu algoritma adımlarında yaşanan kayıp çok çok küçütür. Salt and Pepper yapıldığında ise görüntünün içerdiği bit değerleri bozulmakta ve yaklaşık 0.21 değeri elde edilmektedir. Görüntü temel olarak seçilebilse bile kaba-kuvvet saldırımızda gördüğümüz üzere bu benzerlik değeri görüntüyü başarılı olarak değerlendirebilmek adına yetersizdir fakat görüntü şifreleme algoritmalarında temel amaç görüntünün farkedilebilmesi olduğundan algoritmanın temel işleyişinin güvenilir olduğu söylenebilmektedir. Median Filtering uygulandığında ise benzerlik değeri yaklaşık olarak 0.39 bulunmaktadır. Görüldüğü üzere Salt and Pepper etkisi azaltıldığında benzerlik oranı yaklaşık 2 kat artmaktadır.

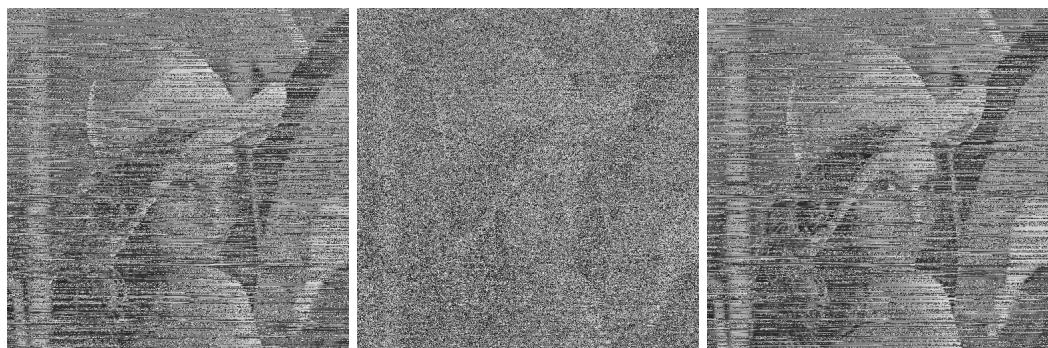
### 7.3 Cropping Saldırısı

Cropping saldırısı, şifrelenmiş görüntünün belirli piksellerinin tamamen karartılması ile deşifrelemenin bozulması esasına dayanmaktadır. Bu saldırında 512x512 lena görüntüsüne 3 farklı karartma işlemi uygulandı ardından bu görüntüler deşifreleme işlemine sokuldu ve çözümlenmiş görüntü elde edildi. Elde edilen bu görüntü orijinal deşifrelenmiş görüntüyle karşılaştırıldı ve analizleri yapıldı.



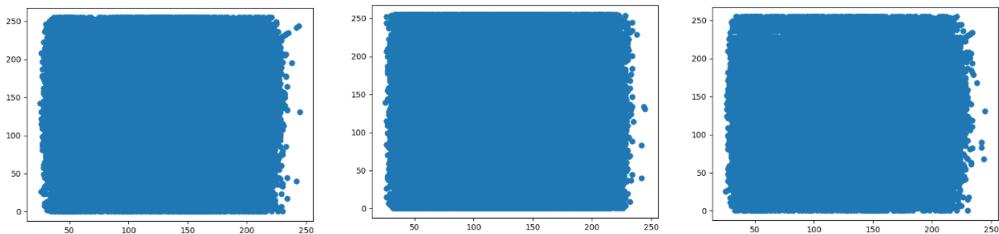
**Şekil 7.12** Cropping Saldırısı Farklı Düzeylerde Gerçekleştirilen 3 Görüntü

Sağlıklı veriler elde edebilmek amacıyla aynı görüntünün şifrelenmiş görüntüsü kullanıldı ve karşılaştırabilmek amacıyla farklı düzeylerde karartıldı. Ardından karartılan bu görüntü dosyalarının KR ve KC dizileri kaydedildi ve deşifreleme işlemi gerçekleştirildi. Deşifrelemenin ardından elde edilen lena görüntüleri aşağıda gösterilmiştir.



**Şekil 7.13** Cropping Saldırısı Ardından Çözümlenen 3 Görüntü

Çözümlenen görüntülerden anlaşılacağı üzere karartma işlemi ne kadar alan kaplarsa çözümlenen görüntünün netliği o kadar bozulacaktır ve karartma işleminin konumunun bu noktada herhangi bir önemi yoktur. Bir sonraki adımda bu görüntülerin orijinal deşifrelenmiş görüntüyle olan benzerliklerinin grafiği ve değerleri gösterilmiştir.



**Şekil 7.14** Cropping Saldırısı Ardından Çözümlenen 3 Görüntünün Orijinal Çözümlenmiş Görüntü ile Benzerlik Grafikleri

Grafikleri gösterilen görüntülerin benzerlik değerleri sırasıyla 0.3061, 0.1097 ve 0.3421 olarak hesaplanmıştır. Yukarıda bahsedildiği ve görüldüğü üzere 1. ve 3. görüntülerin orijinal çözümlenmiş görüntüye olan benzerliği 2. görüntüye oranla yaklaşık 3 kat daha fazladır. Bu noktada saldırında karartılan alan ne kadar fazla yer kaplıyor ise çözümlenen görüntünün de orijinal görüntüye olan benzerliği o kadar azalacaktır sonucu çıkarılabilir mektedir.

Cropping saldırısı, büyük alanları kararttığında görüntünün değerlerini tamamen bozabilmekte ve görüntünün anlaşılması çok büyük ölçüde zorlaştırabilmektedir. Yapılan testlerde görüntünün yaklaşık %15-%20 lik bir kısmı karartıldığında deşifrelenen görüntünün rahatlıkla seçilebildiği ve orijinal görüntüye yaklaşığı fark edilebilmektedir. Yaklaşık %45 i karartılan görüntüde ise görüntü oldukça bozulsa bile orijinal görüntüdeki hatlar fark edilebilmektedir. Bu sonuç algoritmanın güvenilirliğini oldukça desteklemektedir çünkü bir görüntünün verilerinin %20-%40 oranında kaybetmesine rağmen orijinal görüntüyle benzerlik göstermesi görüntü şifreleme algoritmalarında yüksek oranda sağlanabilen bir durum değildir.

# 8

## Performans Analizi

Bu bölümde şifreleme-deşifreleme aşamaları ve yapılan saldırının sonuçları analiz edilecektir.

### 8.1 Şifreleme - Deşifreleme Aşamaları

Proje Intel Core i7-7700HQ 2.80 GHz İşlemci, 16 GB Ram ve Windows 10 İşletim Sisteminde gerçekleşmiştir. 512x512 çözünürlükte bir görüntünün şifreleme ve deşifreleme aşaması yaklaşık olarak 5.43 saniye sürmüştür. 16x16 ve 32x32 çözünürlük için 0.28 saniye, 64x64 çözünürlükte bir görüntü için ise 0.36 saniye sürmüştür. Verilerden görüldüğü üzere görüntünün çözünürlüğü arttıkça yapılacak iş arttıından programın çalışma süresi de artmıştır. Daha performanslı bir işlemci kullanılır ise programın çalışma süresi de azalacaktır. Aşağıda bahsedilen çözünürlükteki görüntülerin şifreleme-deşifreleme aşamaları sonucu geçen süre gösterilmiştir. Farklı çözünürlükteki görüntüler aynı görüntülerin küçültülmesi sonucunda elde edilmiştir.

Çözünürlük	Geçen Süre	Orijinal Boyut	Deşifrelenen Boyut	Şifrelenen Boyut
<b>512 x 512</b>	5.43 sn	462 kB	148 kB	256 kB
<b>64 x 64</b>	0.36 sn	11.6 kB	3.12 kB	4.12 kB
<b>32 x 32</b>	0.28 sn	3.23 kB	985 Byte	1.09 kB
<b>16 x 16</b>	0.28 sn	977 Byte	340 Byte	340 Byte

**Tablo 8.1** Farklı Çözünürlükteki Görüntülerin Şifrelenme Süreleri

Her bir görüntünün şifreleme süresi hesaplanıp ortalama bir değer elde edilmiştir. Geçen süre hesaplanırken python kütüphanelerinden biri olan time kütüphanesi kullanılmış ve program sonucunda terminalde geçen süre gösterilmiştir.

```
lena_64.png Dosyasının Şifreleme ve Deşifreleme Süresi: 0.36162710189819336
```

**Şekil 8.1** 64x64 Görüntü için Hesaplanan Program Süresi

## 8.2 Kaba-Kuvvet Saldırı Aşamaları

Kaba-kuvvet saldırısında ilk saldırısı senaryomuzu  $16 \times 16$  lena görüntüsü üzerinden gerçekleştirdik. Benzerlik değerinin 0.25 değerinden küçük olduğu sürece döngünün devam etmesini istedik ve 15.764. adımada 0.26 benzerlik değerine ulaştığında program sonlandı.

```
----- Attack ----- 15764
----- Cracked Matrix:
[[210 141 83 180 223 171 94 128 225 180 192 176 18 61 85 191]
 [221 224 15 105 40 207 57 130 235 73 144 144 200 144 28 7]
 [ 45 213 50 215 188 63 134 183 161 29 5 66 51 209 189 14]
 [ 79 254 157 177 142 106 104 175 154 152 93 100 150 134 37 249]
 [222 204 198 254 76 87 145 241 136 230 32 174 239 72 43 231]
 [ 57 60 132 111 49 58 228 222 151 104 70 199 45 105 166 56]
 [ 55 110 21 2 223 34 186 207 200 242 201 175 70 180 103 242]
 [ 24 61 109 99 14 55 1 51 30 140 99 235 128 195 154 136]
 [231 73 24 53 174 62 106 235 213 115 174 225 18 99 93 77]
 [ 76 87 211 21 230 41 130 92 218 79 62 37 38 31 224 221]
 [ 20 27 249 107 194 234 200 129 166 104 178 10 45 30 167 253]
 [246 95 239 26 89 31 192 85 159 183 228 130 200 229 166 197]
 [124 53 130 222 17 136 0 68 15 124 210 1 203 191 204 62]
 [242 225 6 59 62 219 1 42 127 192 46 112 238 98 146 52]
 [150 194 156 50 3 91 84 66 53 215 207 74 28 57 111 244]
 [242 241 192 72 20 61 113 79 109 204 107 61 153 67 238 111]]
0.25813953445246984
```

Şekil 8.2 Saldırı Sonucu Oluşan Matrix ve Benzerlik Değeri

Çözünürlük boyutu çok küçük olduğundan net bir benzerlik söz konusu olmadı fakat benzerlik değerinden anlaşılabilceği üzere kaba-kuvvet saldırısı belirli bir seviyede algoritmanın güvenliğini tehlike altına almıştır.



Şekil 8.3 Deşifrelenmiş ve Saldırı Sonucu Oluşan Görüntü

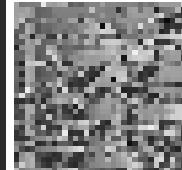
$32 \times 32$  senaryosunda ise toplam 30.661 deneme gerçekleştirilmiş ve 0.45 benzerlik değeri elde edilmiştir. Bu senaryoda  $16 \times 16$  görüntü senaryosunda 0.25 benzerliğin yeterli olmadığı farkedilmiştir ve bu değer 0.40'a çekilmiştir.

```

----- Attack ----- 30661
---- Cracked Matrix:
[[161 118 123 ... 154 155 201]
 [158 156 165 ... 127 180 46]
 [160 167 159 ... 92 57 50]
 ...
 [ 47 150 164 ... 210 90 77]
 [112 205 164 ... 170 112 67]
 [187 124 136 ... 104 69 77]]
0.4507480420796601

Correlation Between Decrypted and Original Matrix: 0.4507480420796601

```



**Şekil 8.4** Program Terminal Çıktısı ve Saldırı Sonucu Oluşan Görüntü

Görüntünün 0.25 değerine kıyasla deşifrelenmiş haline daha çok benzediği ve görüntünün boyutları arttığı için yapılan deneme sayısının neredeyse 2 katına ulaştığında sonuç verebildiği gözlenmiştir.

Programın 64x64 çözünürlüklü görüntü senaryosunda yapılan 78.241 deneme sonucunda 0.25 benzerlik oranı elde edilmek istenmiş fakat herhangi bir sonuç alınamamıştır.

Kaba-kuvvet saldırısının senaryolarından anlayabileceğimiz üzere görüntünün çözünürlüğü arttığında üretilmesi gereken dizi boyutları ve bunun sonucunda yaklaşılması gereken değer sayısı artmaktadır. Değerler rastgele üretildiği ve her adımda değiştiğiinden orijinal 512x512 çözünürlükte bir görüntünün bu yöntemle kırılabilmesi neredeyse imkansızdır. Bu yüzden incelediğimiz Rubik Küpü Şifreleme Algoritmasının Kaba-Kuvvet Saldırılarına karşı dayanıklı olduğu söylenebilir.

### 8.3 Salt and Pepper Saldırı Aşamaları

Salt and Pepper saldırı aşamaları, salt and pepper eklenme ve kaldırılma fonksiyonu dışında algoritmanın orijinal şifreleme ve deşifreleme aşamalarını içermektedir.

Bu program çalıştırıldığında 4 adet çıktı vermektedir, bu çıktılar sırasıyla.

- Şifrelenmiş Görüntü Dosyası.
- Şifrelenmiş Görüntüye Uygulanan Salt and Pepper İşlemi Sonucu Oluşan Dosya.
- Salt and Pepper Uygulanmış ve Ardından Deşifrelenmiş Görüntü Dosyası.
- Salt and Pepper İşlemi Sonrasında Deşifrelenmiş Görüntüden Salt and Pepper Kaldırılmış Görüntü Dosyası.

Terminal çıktısında ise bu 3 dosyanın birbirleriyle olan benzerlik değerleri ve yapılan bütün işlemlerin tamamlanma süresi gösterilmektedir.

lena0.9.png Dosyasının Salt and Pepper İşlemi Gerçekleştirilme Süresi: 11.127048969268799

**Şekil 8.5** Salt and Pepper Saldırısı Gerçeklenme Süresi

#### 8.4 Cropping Saldırı Aşamaları

Cropping saldırı aşamaları, şifrelenen görüntünün bir görüntü düzenleme programı ile istenilen piksellerinin karartılması ve ardından algoritmanın orijinal deşifreleme işlemlerinin uygulanmasıyla çözümlenmiş görüntünün elde edilmesi esasına dayanır. Bu adımların ardından elde edilen görüntü orijinal deşifrelenmiş görüntü ile benzerlik programında çalıştırılır ve benzerlik grafiği ile benzerlik değeri elde edilir.

Bu işlemler, özellikleri verilen sistemde ve 512x512 lena görüntüsünde yaklaşık 2.5 saniye sürmektedir. Görüntünün karartılması manuel bir işlem olduğundan işlem zamanı hesaplanırken hariç tutulmuştur.

# **9**

## **Sonuç**

---

Bu projede 2011 senesinde ortaya konulan Rubik Küpü Şifreleme algoritmasının gerçekleştirilmesi ve kripto-analizinin yapılması amaçlanmıştır. Makalede anlatılan algoritma tarafımızca anlaşılmış ve özgün olarak koda dökülkerek gerçekleştirilmiştir. Kripto-analiz aşamasında üç farklı saldırı senaryosu belirlenmiş ve tarafımızca kodu yazılarak bu senaryolar özgün olarak gerçekleştirilmiştir. Kaba-kuvvet saldırı senaryosunda kullanılan kaba-kuvvet saldırısının sonuçları algoritmanın bu saldırıyla karşı dayanıklı olduğu bilgisini vermiştir. Salt and Pepper ve Cropping Attack saldıruları senaryolarında elde edilen veriler incelendiğinde ise algoritmanın bu ataklara dayanabildiği ve orijinal görüntü ile yaklaşık görüntüleri gösterebildiği görülmüştür. Yapılan 3 saldırı verileri toplanarak sonuç analizi yapıldığında seçilen ve gerçekleşen algoritmanın güvenli olarak kabul edilebildiği ve saldırı testlerinin başarılı sonuçlandığı söylenebilmektedir.

## Referanslar

---

- [1] K. Loukhaoukha, J.-Y. Chouinard, and A. Berdai, “A secure image encryption algorithm based on rubik’s cube principle,” 2011.
- [2] S. Y. Fatih Özkaynak Ahmet Bedri Özer, “Cryptanalysis of a novel image encryption scheme based on improved hyperchaotic sequences,” 2012.
- [3] H. Fan and M. Li, “Cryptanalysis and improvement of chaos-based image encryption scheme with circular inter-intra-pixels bit-level permutation,” 2017.
- [4] M. M. Chunlin Song Sud Sudirman, “A spatial and frequency domain analysis of the effect of removal attacks on digital image watermarks,” 2010.
- [5] A. M. I. S Geetha P Punithavath, “A literature review on image encryption techniques,” 2018.
- [6] M. J. Shishir Sharma A.V. Subramanyam, “Anti-forensic technique for median filtering using l1-l2 tv model,” 2016.

# Özgeçmiş

---

## BİRİNCİ ÜYE

**İsim-Soyisim:** Ömer Buğrahan ÇALIŞKAN  
**Doğum Tarihi ve Yeri:** 02.02.1999, Erzincan  
**E-mail:** l1117076@std.yildiz.edu.tr  
**Telefon:** 0506 366 87 31  
**Staj Tecrübeleri:** -

## İKİNCİ ÜYE

**İsim-Soyisim:** Ömer Aras KAPLAN  
**Doğum Tarihi ve Yeri:** 01.04.1998, İstanbul  
**E-mail:** l1117039@std.yildiz.edu.tr  
**Telefon:** 0541 534 31 88  
**Staj Tecrübeleri:** -

## Proje Sistem Bilgileri

**Sistem ve Yazılım:** Windows İşletim Sistemi, Python  
**Gerekli RAM:** 4GB  
**Gerekli Disk:** 128MB